REPLICA CREATION ALGORITHM FOR DATA GRIDS

MOHAMMED KAMEL MADI

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2012

< INSERT PERAKUAN KERJA TESIS / DISERTASI

(CERTIFICATION OF THESIS / DISSERTATION)>

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to :

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUMCollege of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Sistem grid data merupakan prasarana pengurusan data yang memudahkan pencapaian dan perkongsian data yang banyak, sumber storan, dan perkhidmatan pemindahan data yang boleh merentasi lokasi teragih. Tesis ini mengemukakan satu algoritma baru yang dapat mempertingkatkan prestasi capaian data pada grid data dengan mengagihkan salinan data yang berkenaan. *Data Replica Creation Algorithm (DRCM)* meningkatkan prestasi sistem grid data dengan mengurangkan masa pelaksanaan kerja dan memanfaatkan cara terbaik penggunaan sumber grid data (ruang storan dan jalur lebar rangkaian). Algoritma semasa tertumpu kepada bilangan capaian untuk menentukan fail mana yang patut disalin dan lokasi di mana ia patut diletakkan, yang mengabaikan kebolehan sumber. *DRCM* sebaliknya mengambil kira perspektif pengguna dan sumber; meletakkan salinan tersebut di tempat strategik yang memberikan kos pemindahan yang paling rendah. Algoritma ini menggunakan tiga strategi: *Replica Creation and Deletion Strategy (RCDS), Replica Placement Strategy (RPS),* dan *Replica Replacement Strategy (RRS). DRCM* dinilai menggunakan simulasi rangkaian (OptorSim) berdasarkan metrik prestasi terpilih (purata masa pelaksanaan kerja, penggunaan rangkaian yang cekap, purata penggunaan storan, dan penggunaan elemen pengkomputeran), senario, dan topologi. Hasil kajian mendapati masa pelaksanaan DRCM adalah lebih baik dengan penggunaan sumber yang lebih rendah berbanding pendekatan yang sedia ada. Penyelidikan ini telah menggabungkan beberapa strategi penyalinan dalam satu algoritma yang meningkatkan prestasi grid data, berkeupayaan membuat keputusan serentak ke atas penciptaan atau penghapusan sekurang-kurangnya satu fail dalam pembuatan keputusan yang sama. Di samping itu, kriteria tahap-kebergantungan-antara-fail telah diguna dan diintegrasikan dengan model pertumbuhan/penyusutan eksponen untuk memberi suatu penilaian fail yang tepat.

Kata Kunci: Penyalinan data, Grid data, Pembuatan salinan, Penilaian fail, Penempatan salinan.

Abstract

Data grid system is a data management infrastructure that facilitates reliable access and sharing of large amount of data, storage resources, and data transfer services that can be scaled across distributed locations. This thesis presents a new replication algorithm that improves data access performance in data grids by distributing relevant data copies around the grid. The new Data Replica Creation Algorithm (DRCM) improves performance of data grid systems by reducing job execution time and making the best use of data grid resources (network bandwidth and storage space). Current algorithms focus on number of accesses in deciding which file to replicate and where to place them, which ignores resources' capabilities. DRCM differs by considering both user and resource perspectives; strategically placing replicas at locations that provide the lowest transfer cost. The proposed algorithm uses three strategies: Replica Creation and Deletion Strategy (RCDS), Replica Placement Strategy (RPS), and Replica Replacement Strategy (RRS). DRCM was evaluated using network simulation (OptorSim) based on selected performance metrics (mean job execution time, efficient network usage, average storage usage, and computing element usage), scenarios, and topologies. Results revealed better job execution time with lower resource consumption than existing approaches. This research contributes replication strategies embodied in one algorithm that enhances data grid performance, capable of making a decision on creating or deleting more than one file during same decision. Furthermore, dependency-level-between-files criterion was utilized and integrated with the exponential growth/decay model to give an accurate file evaluation.

Keywords: Data replication, Data grid, Replica creation, File evaluation, Replica placement.

Acknowledgement

First, I would like to express my utmost gratitude to Almighty Allah for his creation and making me submissive to Him.

I would like to gratefully acknowledge the enthusiastic supervision of my thesis supervisor, Associate Professor Dr. Suhaidi Hassan and my Co-supervisor Dr. Yuhanis Yusof. I could not have imagined having a better adviser and mentor for my Ph.D., and without their inspiration, stimulating suggestions, sound advice, guidance, and active participation in the process of work, I would never have finished

Parts of this work would not have been possible without the active contribution of my colleagues InterNetWorks Research Group. I thank them for the long discussions we had, the papers they helped write, and their valuable feedback; among them are Dr Massudi Mahmuddin, Dr Omar Almomani, Dr Mohamad Kadhum, Dr. Ahmad Suki, Hasbullah, Khuzairi Mohd Zaini, Yaser Miaji, Mahmoud Alshogran, Shahrudin, and many others.

Throughout this work I have met with many great people; among them are Associate Professor Dr. Hassan Bouzahir and Associate Professor Mustafa Kalmun, my deep gratitude to them for their valuable suggestions and comments. Their proactive support, advice, and ideas have been critically essential to this research.

I am also grateful to School of Computing, UUM College of Arts and Sciences, for providing research facilities and related resources that facilitates my study. It was an enjoyable place to work and study.

Finally, I am immeasurably grateful to my family for their kindness and support. They have always been there and encourage me; without their selfless love, I know I could not have successfully completed my doctoral studies.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

AC	Access Cost
ASU	Average Storage Usage
CMS	Compact Muon Solenoid
CE	Computing Element
CE Usage	Computing Element Usage
CDN	Content Delivery Network
DDB	Distributed Database
DMS	Dynamic Maintenance Service
DORS	Dynamic Optimal Replication Strategy
DRCP	Dynamic Replica Creation And Placement
ENU	Efficient Network Usage
EU DataGrid	European data grid
FSR	Fair Share Replication
FL	File Lifetime
FTT	File Transfer Time
FV	File Value
FW	File Weight
GridFTP	Grid File Transfer Protocol
BHR	Hierarchy Based Replication
ISP	Information Service Provider
LALW	Last Access Largest Weight
LFF	Least Frequent Files

LFU	Least Frequently Used
LRU	Least Recently Used
MJET	Mean Job Execution Time
MB	Megabytes
MFS	Modified Fast Spread
MONARC	Model Of Networked Analysis At Regional Centers
MFF	Most Frequent Files
MC	Move Cost
NoA	Number Of Access
ONR	Optimal Number Of Replica
PB	Petabytes
PBRP	Popularity Based Replica Placement
QAC	Queue Access Cost
QAC	Queue Access Cost
QL	Queue Length
RC	Read Cost
RC	Replica Catalogue
RCDS	Replica Creation / Deletion Strategy
RLS	Replica Location Services
RmGrid	Replica Management In Grid
RMS	Replica Management Service
ROP	Replica Optimization Problem
RO	Replica Optimizer

RPS	Replica Placement Strategy
RS	Replica Selection
RB	Resource Broker
RM	Replica Maintenance
RS	Required Space
SBU	Simple Bottom-Up
SDSS	Sloan Digital Sky Survey
SC	Storage Cost
SE	Storage Element
SE Usage	Storage Element Usage
TCP/IP	Transmission Control Protocol / Internet Protocol
TH	Threshold Value
VO	Virtual Organizations
WNs	Worker Nodes
WWW	World Wide Web

CHAPTER ONE
INTRODUCTION

1.1 Introduction

With rapid advances in scientific instrumentation and simulation, scientific data are growing fast in both data size and data analysis complexity. The next generation of scientific applications in domains as diverse as high energy physics, climate modeling, and earth sciences involve the production of large datasets from simulations or large-scale experiments. Analysis of these datasets and their dissemination among researchers located over a wide geographic area requires high capacity resources such as supercomputers, high bandwidth networks, and mass storage systems.

The grid computing [1, 2] paradigm unites geographically-distributed and heterogeneous computing, storage, and network resources and provide unified, secure, and pervasive access to their combined capabilities. Therefore, grid platforms enable sharing, exchange, discovery, selection, and aggregation of distributed heterogeneous resources such as computers, databases, visualization devices, and scientific instruments [3]. Hence also, leads to the creation of virtual organizations [4-6] by allowing geographically-distributed communities to pool resources in order to achieve common objectives. These resources can be divided into computing or storage units that can be accessed or shared by large numbers of remote users. Computing unit or Computational Grid [7] focuses on supplying computing power, while storage unit or data grid focuses on enabling and facilitating reliable access and sharing of data management resources in widely distributed locations.

A data grid [8, 9] is an infrastructure that deals with huge amounts of data to enable grid applications to share data files in a coordinated manner. Such an approach is seen to provide fast, reliable and transparent data access. Nevertheless, data grid creates a challenging problem in a grid environment because the volume of data to be shared is large despite the limited storage space and network bandwidth [10, 11]. Furthermore, resources involved are heterogeneous as they belong to different administrative domains in a distributed environment. It is unfeasible for all users to access a single instance of data (e.g. a data file) from one single organization (e.g. site). This would lead to the increase of data access latency. Furthermore, one single organization may not be able to handle such a huge volume of data by itself.

Motivated by these considerations, a common strategy is used in data grids as well as in distributed systems, and this strategy is known as replication. Replication vouches efficient access without large bandwidth consumption and access latency [12-18]. The replication technique is one of the major factors affecting the performance of data grids [19]. Creating replicas can reroute client requests to certain replica sites and offer higher access speeds. Hence, well-defined replication strategies will smooth data access, and reduce job execution cost [20]. Such a strategy should also be able to deal with dynamic changes in the grid environment, such as dynamic resource availability and access patterns.

Figure 1.1 shows a high-level view example of a worldwide data grid, consisting of computational and storage resources in different countries that are connected by high speed networks. The thick lines show high bandwidth networks linking the major

centers and the thinner lines are lower capacity networks that connect the latter to their subsidiary centers. The data which were generated from an instrument, experiment, or a network of sensors is stored in its principal storage site and is transferred to the other storage sites around the world on request through the data replication mechanism. Users query their local replica catalog to locate datasets that they require. The data may be transmitted to a computational site such as a cluster or a supercomputer facility for processing. After processing, the results may be sent to a visualization facility, a shared repository, or to the desktops of the individual users.

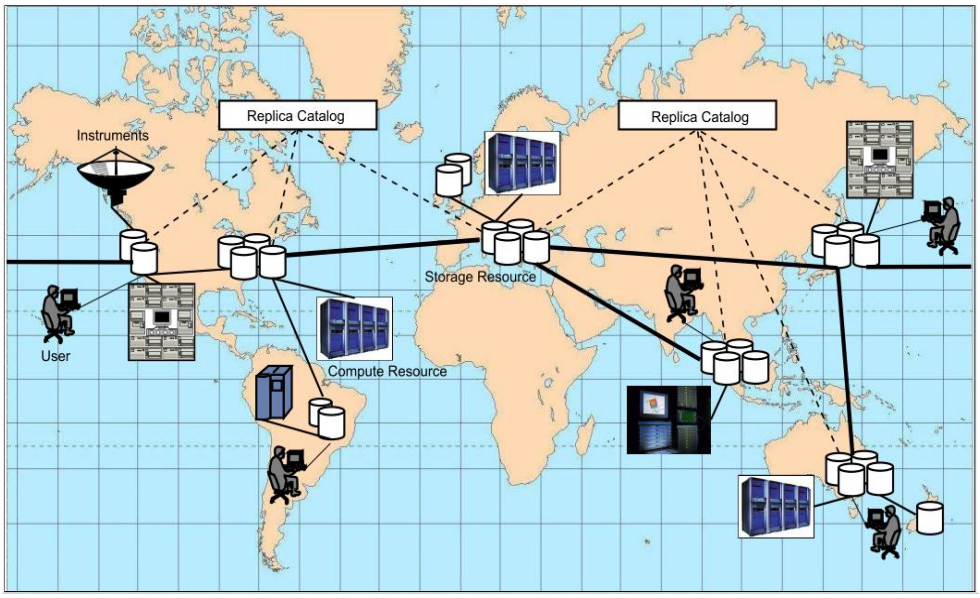


Figure 1. 1: A High-Level view of a data grid [21].

1.2 Stages of Replication Strategies

Data replication is the process of producing multiple copies of data files, and distributing these replicas to Grid sites according to some techniques termed as

replication strategies. This work categorizes replication strategies into two stages, as shown in Figure 1.2:

i. **Replica Creation Stage (RCS):** This is the stage where decisions on which file to replicate and where to locate the newly created replica are identified. Three strategies are embodied at this stage. First is the replication decision strategy that decides which file is to be replicated, second, a replica number strategy that determines the required number of replicas, and third, a replica location strategy that finds suitable places to host the new replicas. This thesis is on RCS as studied by several other researchers [13, 18, 20, 22-24].

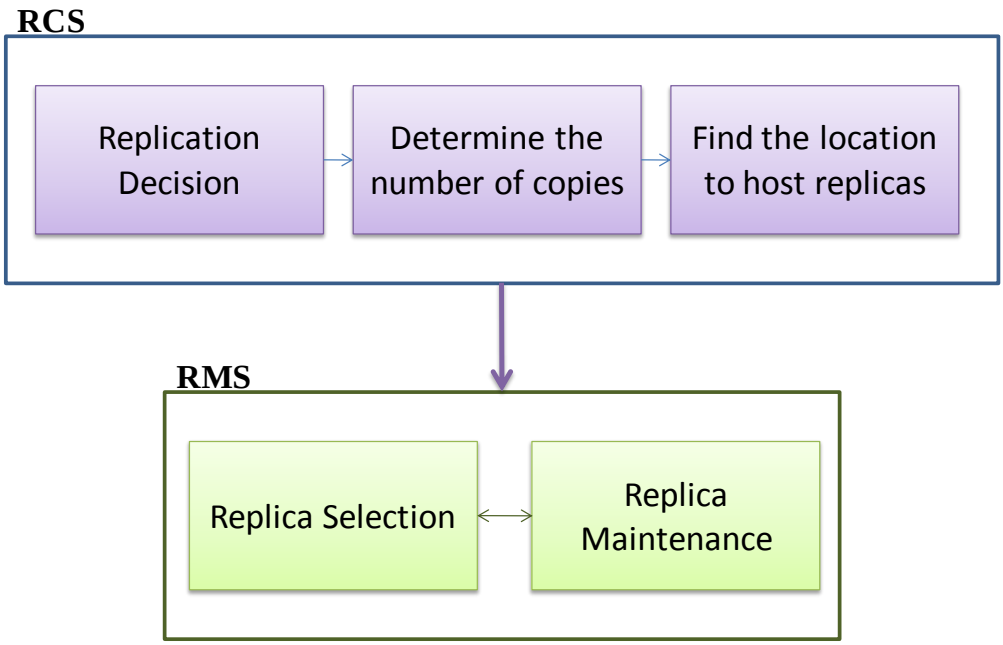


Figure 1. 2 Stages of Replication Strategies

ii. **Replica Management Stage (RMS):** Having distributed the replicas to various locations, they need to be monitored. As a result of dynamic changes in the data grid environment, some of the replicas need to be relocated. The main aim of RMS

is to relocate the replicas by distributing them to new locations based on the adopted mechanism [25-29].

1.3 Problem Definition

Replication can be motivated by two issues, availability of data (fault tolerance) and system performance [30, 31]. In a data grid, the high level of reliability [31, 32] of the main data storage sites makes fault tolerance less of an issue, while the large file sizes increase the file access times of grid jobs. Therefore, performance becomes the main motivation for replication in data grids.

In the context of the data grid, increasing the performance of the system can be achieved by improving the overall resource usage, which includes network and storage resources [33]. Improving network resource usage is achieved by good utilization of network bandwidth that is considered as an important factor affecting job execution time [34]. Meanwhile, improving storage resource usage is achieved by good utilization of storage space usage [35].

Performing data replication introduces additional problems: the decision of replication must be wisely made (identifying the appropriate data file to be replicated), replicas must be properly located, their numbers must be properly determined, their lifetime must be managed properly, and the related storage and resources must be utilized efficiently. To sum up, data replication process has to take into account both users' and system's perspectives. Even though these problems can

be solved by existing replication algorithms [20, 22, 24, 36-46], existing work require enhancements due to the absence of system's perspective in terms of:

i. **Replication decision making:** the replication is performed (deciding the file to be replicated and the required number of replicas) based on users' perspective, i.e. according to number of access of a file. Therefore, the number of times a system makes replication has a possibility to be increased. As a result, the network usage would be affected, this is because each replication consumes network bandwidth and increases network traffic. Moreover, the replication decision of current works does not involve the deletion process of unwanted replicas in their decision. Thus the storage cost will be increased. In this context, storage cost is the space used to store data. Therefore, increasing the storage cost would lead to less storage availability. According to [47] less storage availability would lead to longer job execution time and larger network usage because only fewer replicas can be accommodated in the data grid, and most files will be read remotely. Furthermore, leaving free space can improve the function of other strategies of the replica creation system. When creating new replica, a storage location must be specified to place this replica; if the storage is full, the system needs more time to delete one of the existing replicas in order to free storage space for the new replica. In addition, the deleted replica may still has a demand which lead the system to create this replica again next time, and this situation may cause the system to be exhausted.

ii. **Replica placement decision:** the current works places the replicas at sites that issue the highest number of request, regardless the current situation of the grid in terms of capabilities of resources (i.e. maximum workload and available storage space), and the places of existing replicas.

1.4 Objectives of the Research

The main goal of this research is to develop a replication algorithm aimed at improving the performance of the data grid in terms of minimizing job execution time, reducing storage cost, and minimizing network bandwidth consumption. In order to achieve this research goal, the research objectives were formulated, which are:

i. To design a data replica creation and deletion algorithm (DRCM), based on exponential growth/decay model with the concept of file dependency and read cost. DRCM considers the capabilities of grid system in terms of workload and distribution of existing replicas. In order to achieve this objective, the following sub-objectives have been formulated:
    a. To formulate a function that determines the importance of a file to both users and system perspective.
    b. To design a replica creation/deletion strategy that determines which replica is to be created or deleted and the appropriate number of replicas.
    c. To design a replica placement strategy that determines the best site to place the replica at or to delete the replica from.

d. To design a replica replacement strategy that identifies the victim file to be replaced by the newly created replica in case of insufficient storage space at the candidate site.

ii. To implement and evaluate the proposed algorithm. Thus the proposed algorithm can be tested and evaluated in order to measure its performance.

1.5 Significance of the research

The proposed replication algorithm can be considered as a long-term strategy that aims at best utilization of grid resources usage, namely reducing storage use and reducing network bandwidth consumption. In other words, this proposed algorithm gives a bird's eye view on all components; in a grid environment, the system designers or system administrators would be interested in this view in order to determine the overall resource requirements and to configure, to monitor, and to control the overall system components.

The proposed replication algorithm is also beneficial for grid users as job execution time is reduced. Users' jobs which are under execution would require data files and the grid system in turn would place the required files (i.e. replicas) as close as possible to the users (i.e. requesting sites).

1.6 Scope of the research

This research focused on replica creation or deletion in the data grid. The following describes the scope of this work:

i. The data used in this research is of read-only type. Thus, this research has not considered the consistency of write types and overheads of update propagation costs in this research.

ii. This research focuses on a tree-like-structured grid model, which reflects the hierarchical structure in grid systems [18, 48, 49]. The hierarchical data grid model is a common architecture used in various research works [18, 48-50].

iii. The modality of data that is used in this work is in the form of structured data, specifically source code data.

1.7 Thesis Layout

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of the background material and establishes the concepts and issues covered in the thesis. In this chapter, a brief critical study and survey of the relevant existing studies are presented.

Chapter 3 describes the tools and methodology used in the investigation. A brief description of the simulator used in this research is given. The chapter presents the grid architecture, grid internals, and the simulator inputs. The performance evaluation metrics that were used as benchmarks to evaluate the proposed algorithm are presented in this chapter as well.

Chapter 4 describes the research solution that is encapsulated in a replica creation algorithm for solving the research problem. The algorithm requirements, design, components, and algorithms of the proposed algorithm are explained. Some numerical examples are provided in order to clearly demonstrate how the proposed algorithm works. This chapter also covers the implementation of the proposed

algorithm, which includes the integration of the new algorithm into the simulation environment, and the required modifications in the coding.

Chapter 5 presents the different scenarios used in the simulation experiments. Results produced from the simulation are discussed and compared with other similar algorithms.

Chapter 6 summarizes the research work, highlights research contributions, and gives direction for future work related to this research.

CHAPTER TWO
BACKGROUND AND RELATED WORKS

This chapter first explores: data grid and the challenges of data grid by illustrating some examples of the growth of data requirements for the scientific applications. Then related data-intensive studies are explored in order to provide an overview of the area and domain of this research. Then data replication strategies of Replica Creation Stage are discussed in details, each strategy discussed with the corresponding related works. The analysis of the features and limitations on the state of the art of replica creation stage strategies is performed.

2.1 Data Grid

The term data grid [51-53] refers to an infrastructure that provides data management services for users in order to access, store, transfer, and replicate data files located within distributed storage media. Moreover, a data grid connects a collection of hundreds of geographically distributed computers and storage resources to facilitate sharing of data, storage resources, and computational power [8, 54].

Through the linking of all these equipment, the Grid can provide a platform through which users can access aggregated computational, storage, and networking resources to execute their data-intensive applications using remote data [55, 56]. It promotes a rich environment for users to analyze data and share the results with their collaborators across institutional and geographical boundaries [21, 57, 58].

2.2 The Challenges in Scientific Grid

The first Grid was conceived by computing science [59, 60]. The scale of scientific experiments has grown so fast that traditional methods of computing used to solve associated problems are now quite inadequate. Scientific experiments such as high-energy physics [61, 62], climate modeling, earthquake engineering [63, 64], bioinformatics [65], and astronomy are generating huge volumes of data which are measured in terabytes and rising to petabytes within just a couple of years [66]. There are many examples that illustrate the spectacular growth of data requirements for scientific applications [67], as will be described in the following sections.

2.2.1 High Energy Physics

The most cited example of massive data generation in the field of High Energy Physics (HEP) [68] is the Large Hadron Collider (LHC), which is the most powerful giant particle accelerator at CERN (the European Organization for Nuclear Research) [69]. HEP consists of four main experiments namely ALICE [70], ATLAS [71], CMS [72-74], and LHCb [61], which are designed to understand the fundamental particles of matter and the forces acting between them. HEP experiments will produce several petabytes of raw and derived data that will be accessed from different centers around the world through very heterogeneous computational resources. The raw data are generated at a single location (CERN) where the accelerator and experiment are hosted, but the computational capacity required to analyze them implies that the analysis must be performed at geographically distributed centers. In practice, CERN's experiments are

collaborations among thousands of physicists from about 300 universities and institutes in 50 countries.

In order to produce a suitable computing model to handle the needs of CERN's experiments, the MONARC [75] (Model of Networked Analysis at Regional Centers for LHC Experiments) project was initiated in 1998 [31]. The MONARC model, based on distributed computing following a hierarchical architecture, centered on CERN. A multiple tier system was proposed, which are composed as follows:

The central **Tier-0** is hosted by CERN, where all the data are produced and where the experiments are situated. The original raw data are recorded and archived here.

Tier-1 consists of 10 sites that act as regional centers for the country or region in which they are situated and each of which will serve some or all of the experiments.

Tier-2 sites will each serve some geographical area (perhaps a region within a country). The role of Tier-2 sites is to provide computing resources and some storage capacity.

Tier-3 layer consists of the computing facilities at universities and different institutes, which are conceived as relatively small structures connected to reference Tier-2 Centers.

Tier-4 centers are users' personal desktops.

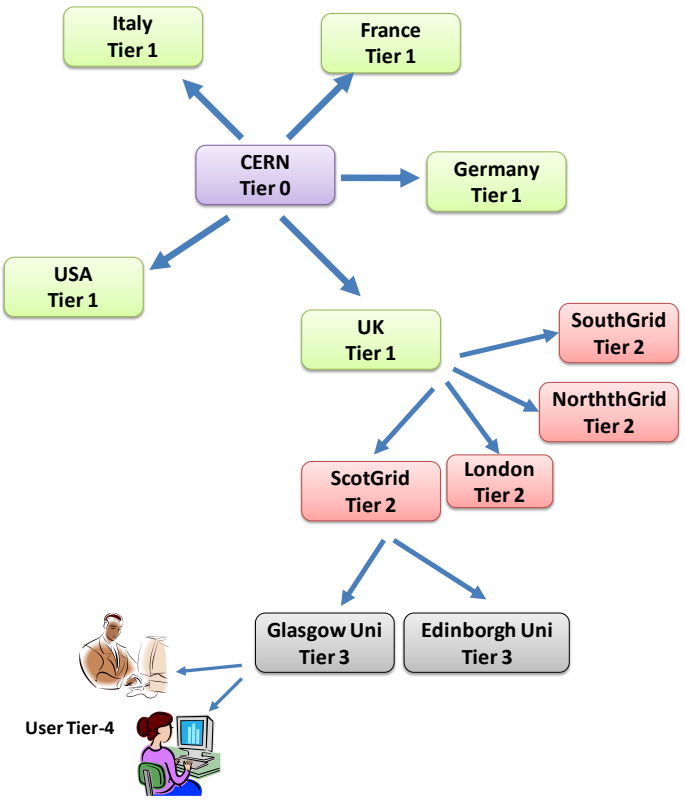


Figure 2. 1: CERN replication scheme in a hierarchy form

2.2.2 Other Data-Intensive Disciplines

There are many examples of the growth of data requirements for scientific applications. The following are some of the more significant applications which have gathered more attention from the research community.

Climate Modeling

Another example of science that faces large quantities of data is climate model computations [8]. Climate modeling requires long duration simulations and generates very large files that are needed to analyze the simulated climate [76]. These

simulations, however, will produce tens of petabytes of output in future and if this output is to be useful it must be distributed to climate researchers at various institutions.

Bioinformatics

Genomics require programs such as genome sequencing projects, which produce huge amounts of data. The analysis of these raw biological data requires very large computing resources. Bioinformatics [65] involve the integration of computers, software tools, and databases in an effort to address these biological applications, since genome sequences provide copious information about species from microorganisms to human beings. The analysis and comparison of genome sequences are necessary for the investigation of genome structures which is useful for the prediction about the functions and activities of organisms.

Astronomy

Another data-intensive application in the astronomy field is the Sloan Digital Sky Survey (SDSS) [77] which aims to map in detail one quarter of the entire sky and determines the positions and absolute brightness of more than 100 million celestial objects. It will also measure the distances to more than a million galaxies. SDSS and other astronomy applications are performed in several regions of the electromagnetic spectrum and produce an enormous amount of data.

There are many other examples which could be drawn from chemistry [78], engineering [79], and earth science [9]. Suffice to say that science in general is

facing a flood of data as technology develops and that in many cases, grids are seen as a viable solution to address these problems.

2.3 Data Grid Layered Architecture

The layers outlined in Figure 2.1 [13, 52] represents different components that make up the data grid infrastructure. Components at the same level can cooperate to offer certain services, and components at higher levels use components offered at lower levels.

The applications layer provides services and access interfaces for a specific community. These services invoke services provided by the layers below and customize them to suit the target domains, such as high energy physics, biology, and climate modeling.

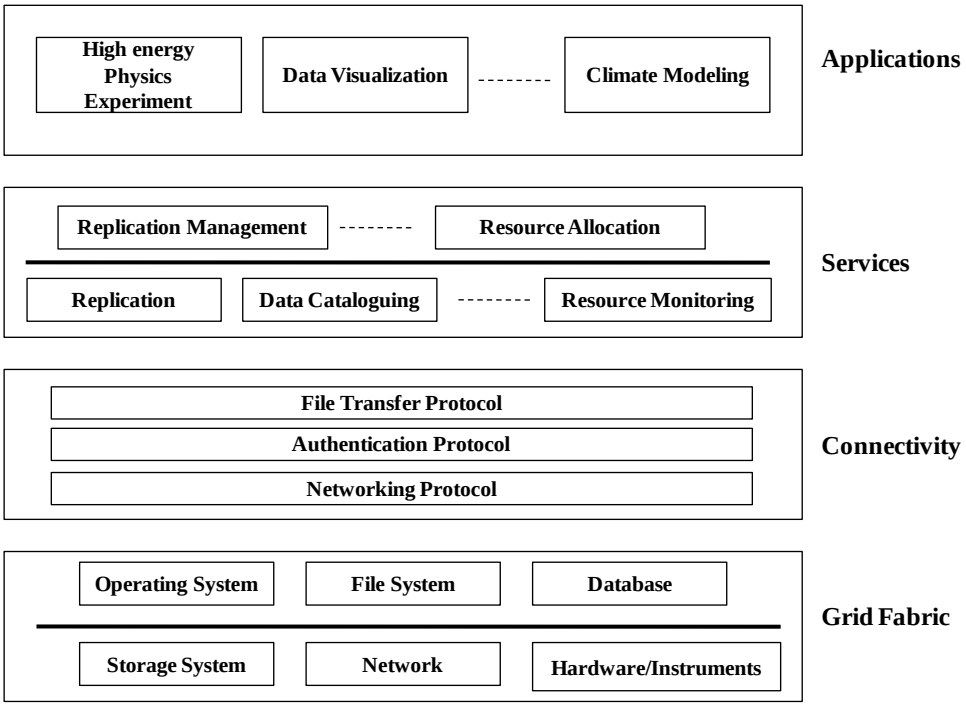


Figure2. 2: Overview of data grid architecture [13, 52]

The services layer is divided into two sub-layers: the high-level sub-layer and the low-level sub-layer. The high-level sub-layers are the services located in the upper layer such as replication management, replica selection optimization, and resource allocation. The high-level sub-layers make use of the low-level sub-layers in order to improve the service quality for users. Replication management service manages the number of replicas and their locations in the grid sites in order to optimize the grid resource usage. However, the replica selection service provides the best replica location for the users or the jobs under execution. The low-level services at the same layer provide services to the upper level such as replication, data cataloguing, and resource monitoring. The data catalogue service provides a number of services such as record all replicas and their physical locations on the grid sites, register the newly created replicas, and delete the replicas from the registry that has been decided to be deleted by the replication management service. The replication service is different from the replication management service. The replication management service decides, and the replication service executes what has been decided by the replication management service. Once the replication management decides to create a new replica, the replication service creates a new copy of the specified file and uses data transfer service to move the copy (replica) to the underlying site location that is determined by the replication management service.

The connectivity layer consists of protocols used to query resources in the grid fabric layer and to conduct data transfers between them. These protocols are built on core protocols for communication such as TCP/IP and file transfer protocols (for example

GridFTP). The grid fabric consists of software and physical hardware components such as computing and storage resources.

2.3.1 Related Data-Intensive Environments

A data intensive computing environment consists of applications that produce, manipulate, or analyze data in the range of hundreds of MegaBytes (MB) to PetaBytes (PB) and beyond [80, 81].

The most related distributed data-intensive research areas that share a few of the characteristics and replication function with the data grid have been briefly studied in this research, which are described briefly in the proceeding sections.

Content Delivery Network

A Content Delivery Network (CDN) [82, 83] consists of a collection of (non-origin) servers that deliver specific content, such as web pages, streaming media, and real-time video, on behalf of the content provider [84]. There are two main structures of CDN, namely commercial architecture and academic architecture. The commercial architecture is based on a client-server network such as Akamai [85]. However, in academic architecture the CDN is based on a Peer-to-Peer architecture that depends on the users (content providers) themselves, whom have to become a part of the voluntary of servers.

Distributed Database

A distributed database (DDB) [86, 87] is a logically organized set of data stored at different sites of a computer network. Each site has a degree of autonomy, is capable

of executing a local application, and also participates in the execution of a global application [21]. A DDB has emerged for fulfilling the need of large organizations to increase data availability and reliability by replicating the database into different sites [87]. Data consistency is a major concern of database management because most transactions are of type data updates rather than read-only procedures as in the data grid [88].

World Wide Web (WWW) Web Applications

WWW has become a ubiquitous media for content sharing and distribution [89]. Applications using the Web spans from small business applications to large scientific calculations. In Web applications, data replication is used in the form of Web caching [90]. The most frequently used web documents are replicated and stored on servers where the most requested users are nearby in order to reduce web response latency and server loads. However, web caching deals with web page documents which are very small in size in relation to the data size used in data grids.

In conclusion, it may be agreed upon that data grids share many characteristics with other types of data-intensive domains. However, data access type could be dissimilar in each environment. Most well-established data grid applications are used in scientific and engineering fields, and they primarily require large read-only data sets, so replica synchronization is not a critical concern in data grids [51]. However, synchronization and consistency are the key features for DDBs replication or Web caching. A DDB typically uses short transactions while scientific ones tend to be long, so the issues are very different [91, 92]. P2P content sharing networks are

mostly read-only environments and write operations occur when an entity introduces new data into the network or creates copies of existing data. CDNs are almost exclusively read-only environments for end-users and updating of data happens at the origin servers only [21, 88].

Moreover, there are some characteristics of data grids that differ from the relative data-intensive applications and environments, some of which are: heavy computational requirements, wider heterogeneity, large data sizes exhibiting special data localities, the presence of Virtual Organizations (VO) [4, 93], different administrative domains, and the emergence of scientific research that produces huge volumes of read-only data files to be shared by many grid users across the globe.

2.4 Replication in Data Grids

One of the principle goals of data grids is to improve transparent access to globally distributed data, making data access and location as easy as if it is occurring on a local computer [14]. Optimization of data access can be achieved via data replication [94, 95], whereby identical copies or replicas of data are generated and stored at distributed sites. Data replication increases the data availability and reliability for the users and decreases the job execution time, but on the other side the replication increases the storage cost, and affects the network bandwidth consumption either positively or negatively. The replication strategies influence the network bandwidth positively when the number of replicas are balanced and distributed across grid sites efficiently. However, the replication strategies affect the network bandwidth negatively when the numbers of replicas are not proportional to the appropriate replica demand.

This section explores the current grid systems and middleware architecture and features by highlighting the replication mechanism.

2.4.1 Storage Resource Broker

SRB [96] is a client- server middleware that provides a management system for data replica and a uniform single interface. SRB manages heterogeneous distributed data storage to allow users to access files and database seamlessly. The unified view of the data files stored in disparate media and locations are provided, and transparent to the users so that the dispersed data appears to the user as stored locally [97]. Data replication in SRB is applicable if the data is required to be much closer to the user [98]. Replicas can be created using SRB or from outside the system and several forms of data replication are possible.

2.4.2 Grid Data Farm

Grid data farm [99] is defined as a group of physical files that distributed across grid sites and appear to the user as a single logical file system that stored in the form of fragments. Individual fragments can be replicated and managed in order to provide service to the data-intensive applications. While executing a program, the process scheduler dispatches it to the site that has the segment of data that is required by the program. If the sites that house the required data are overloaded, the file system creates a replica of the required fragment on another site.

2.4.3 Globus Toolkit

As defined and explained by Ian Foster [100] Globus is:

- A *community* of users who collaborate on sharing of grid resources across cooperate, institutional, and geographic boundaries. Globus also is a community of developers for the development of open source software, and related documentation for building grids and grid based applications for distributed computing and resource federation.
- The *infrastructure* that supports this community such as: code repositories, interface, protocols, email lists, and problem tracking systems.
- The *software* itself, which consists of a set of libraries and programs for solving common problems that occur when building distributed system services and applications.

The Globus data grid architecture [101, 102] is divided into two main layers: high-level services and core services, as shown in Figure 2.3. The hierarchical organization explains the possibilities for using the core services to build the high-level service, so that many data management services and complex storage management systems such as Storage Resource Broker (SRB), can share common low level mechanisms. The services that Globus offers are: Security, Information Services, Resource Management, and Data Management. The Information Services provide information about the status of grid resources. The Resource Management uses information from Information Services to enable users to access available resources and to allow the system to schedule resource allocations. The Data Management provides the ability to access and manage data and data resources on the grid [13]. The Globus toolkit provides several components to move, copy, and locate data.

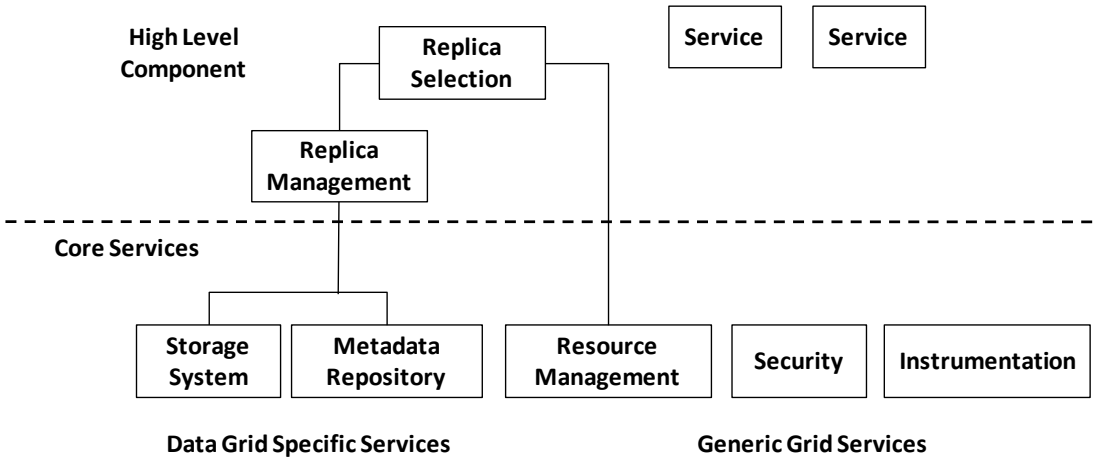


Figure 2. 3: Globus data grid architecture

Typical usage scenarios of Globus and hence the proposed replication algorithm is in High Energy Physics (HEP) applications. High Energy Physics (HEP) data management requires very large amounts of both processing power and data storage.

The four experiments of the Large Hadron Collider (LHC) will accumulate of the order of 5-8 petabytes of raw data per year. In addition, during the preparation phase prior to the start of LHC data taking, a similar order of magnitude of simulated data will be required to design and optimize the detectors. Each LHC experiment will form a single Virtual Organization (VO), comprising of the order of 2000 scientists from over 50 countries. HEP community seeks to take advantage of the distributed nature of computing grids to provide physicists with the best possible access to both simulated and real LHC data, from their home institutes. Data replication and management is hence considered to be one of the most important aspects of HEP computing grids. The task of replicating LHC data to the various collaborating

institutes within a VO will be handled by Data Management services of Globus, such as DRCM.

As mentioned in Chapter 1, data replication can be logically separated into two core and sequential stages: Replica Creation Stage, and Replica Management Stage. These are outlined in the next subsection.

2.4.4 Replica Creation Stage

Replica Creation Stage (RCS) encompasses three main phases:

a) *Replication Decision*: Firstly, given the trigger condition, the replication strategy must be decided, in other words, whether to replicate or delete the file. The replication decision is taken based on the algorithm itself.

b) *Determination of the number of replicas:* Next, once the file to be replicated or deleted is known, the number of replicas should be determined.

c) *Replica placement:* Lastly, the replication strategy should find the new places to host the replicas, or the places from where to delete the replicas.

2.4.5 Replication Decision

In order to perform a replication, a replication trigger must be identified. In general there are two types of triggers that can be considered:

Trigger on file request

When the Storage Element of a site is requested for a file which it does not store, this could trigger a replication strategy. This kind of strategy is also called an unconditional strategy where replication is performed for every request. The most

well-known replacement policies used commonly in operating systems are: Least Recently Used (LRU) and Least Frequently Used (LFU) [103], which are used in page replacement to free the storage space for more important data. LRU and LFU are examples for this kind of replication strategy that is deployed in data grids [18]. In the LRU strategy, the requested site caches the required replica, and if the local storage is full or the current free space is insufficient for the required replica, the least important (victim) replica should be determined and deleted in order to free storage. The victim replica in LRU is the replica that has the maximum period of time between the current time and the last time the replica was requested. However in LFU, the victim replica is the replica that has the least number of requests, or also known as the least popular replica.

Trigger on popularity conditions

Another possible trigger could be a file on some other Storage Element of the site reaching a certain level of popularity. This would require monitoring of all file popularities, perhaps in a central database or by a publish/subscribe method (one Storage Element could subscribe to another one to receive regular updates of its top ten most popular files, for example), and this kind of strategy is also called a conditional strategy.

The process of determining the popularity of a file (identifying which file is to be replicated), may vary from one mechanism to another. The most common characteristic that is widely used to define popularity is the Number of Access (NoA) to the file [20, 36, 37]. NoA stands for the access rate of the file within a certain time

interval. However, determining the certain time interval differs from one mechanism to another. File access pattern analysis has always been employed as a powerful tool to design efficient replication decision [104, 105].

For example, in [22] the authors consider NoA only in the current time interval. The performance of five distinct strategies had been evaluated using simulation framework; 1) Best Client: replica is created for the client who accesses the file the most; 2) Cascading: a replica is created on the path to the best client; 3) Plain Caching: a local copy is stored upon initial request; 4) Caching + Cascading: combines plain caching and cascading; and 5) Fast Spread: file copies are stored at each node on the path to the best client. The evaluation was done using three different kinds of access patterns. Similar to the work undertaken in this thesis, the research does not include consistency issues and the data used in the work was read-only data. The three different access patterns are:

i. Random access pattern, which has no locality in patterns;
ii. Data contain a small amount of temporal locality—temporal locality means that the potential access to the popular file in the past is more than others—where some accessed files are likely to be accessed again; and
iii. Data contain small amount of geographical and temporal locality—the files recently accessed by client are likely to be accessed by nearby clients.

On the other hand some works [20, 24, 39-42] consider NOA in the present and past time intervals, which means that the popularity of the file is determined by analyzing the access history of different time intervals. It has been acknowledged the fact that

files that are requested in the present are apt to be requested in the near future. Therefore, popularity of a file depends on the number of access made to the file by the users. And yet, different calculations are used to determine the popularity of the file. In [24], two replication mechanisms were proposed in the multi-tier architecture for data grids, including Simple Bottom-Up (SBU) and Aggregate Bottom-Up (ABU). The SBU algorithm replicates any data file that exceeds a pre-defined threshold. The main shortcoming of SBU is the lack of consideration to the relationship with historical access records. For the sake of addressing the problem, ABU is designed to aggregate historical records to the upper tier until it reaches the root.

The authors in [20] determined the popularity of the file by analyzing data access history, the average number of access, and computed NoA. Files with NoA values that are greater than the computer average NoA will be replicated. Hence, the order of which files to be replicated depends on the NoA. The larger the NoA, the more popular the file is, and it will be given a higher priority during the replication process. In [41, 106], the average access frequency (freq_{avg}) is calculated as a ratio of the sum of all access frequencies to the total number of files, then the files which have access frequency greater than or equal to freq_{avg} are marked for replication. Nevertheless, these replication strategies do not consider the time period of when the files were accessed. If a file was accessed for a number of times in the past, while none was made recently, the file would still be considered popular and hence it will be replicated. Some economical model-based replica schemes have been proposed.

The authors in [107, 108] use an auction protocol to make the replication decision where the files are evaluated using two prediction functions, namely a binomial-based function and a Zipf-based function. In [109], the evaluation of the files is performed based on number of requests and the existing number of replicas. In [110], the authors suggested that the file must be replicated if it has been requested too many times and there are not enough copies. In other words, the file will be replicated if its average weight exceeds the average weight of the entire grid. Average weight of a file is calculated by dividing number of requests of the underlying file by the number of existing copies, while the average weight of the entire grid is calculated by dividing the total number of requests of the files by the number of existing copies.

Meanwhile, an optimal replication strategy (DORS) has been proposed by [111], where the authors empirically inferred a threshold to decide whether to replicate the file or not. The threshold is represented by the storage system's relative capacity, which is defined as the ratio of the storage size to the total data set sizes (R). When the number of the file's replicas is greater than R, the file will not be replicated, but when the number of the file's replicas is less than R, the file will be replicated.
The work represented in [112] proposed a replication strategy where replicas are automatically increased according to file access. Once the number of accesses of a certain replica is higher than a threshold, it is labeled as "hot data" and replicated.

The algorithms proposed in [37] and [113] are called Last Access Largest Weight (LALW) and Dynamic Replica Creation and Placement (DRCP) respectively, and

both of which tried to solve this problem. The key point of these two algorithms is to give different weights to files having different ages.

The LALW and DRCP algorithms are similar to other algorithms [20, 24, 41] by means of using information on access history to determine the popularity of a file. However, an innovation is included by adding a tag to each access history record of a file. The weight of the record decays to half of its previous weight after a constant time interval. Older access history records have smaller weights; it means that a more recent historical record is more important. An Access Frequency is calculated to represent the importance of access histories in different time intervals.

However, the above approaches (i.e. LALW and DRCP) assume that the decay rate is constant and equals ½, and this means all files decay at the same rate, regardless of the access rate of each one. As a result, the decay rate of weight will be slower. Subsequently the storage element will take time to delete the unwanted files (i.e. the less popular files).

The popularity of the file or the file value is used in two directions: the first direction is to trigger replica creation/deletion strategy as mentioned before. The second direction is to trigger replica replacement strategy, as the less valuable file is replaced by the most valuable file. The difference between replica deletion and replica replacement is that replica deletion is invoked before the replica replacement strategy where the files that have the minimum values are deleted. Meanwhile, the replica replacement strategy is invoked when there is no space for newly created replica in the underlying storage element, and given such a situation, the replica of

low value would be replaced by the replica of higher value. The most well-known replacement policies used commonly in operating systems are: Least Recently Used (LRU) and Least Frequently Used (LFU) [103], which are used in page replacement to free the storage space for more important data. LRU and LFU are examples for this kind of replication strategy that is deployed in data grids [18]. In [114-116], the authors proposed a prediction-based replica replacement algorithm using a two-stage process to evaluate the popularity of a replica. They considered some features such as bandwidth and replica size. The simulation results demonstrated that their algorithm contributed to better grid performance. The work in [117] suggested a replica replacement algorithm based on economic model and opportunity cost, the files have been evaluated using zipf-like distribution prediction model and then weighted using the file transfer cost model. If the needed replica has a higher weight than the replica with the lowest weight in local storage, that file will be deleted and the new replica will be transferred into the local site. In [111], the authors proposed a replacement policy that determines the victim file using two kinds of evaluations.

Firstly by evaluating the replica's access frequency using the half-life principle that is used in [37, 113], and secondly by evaluating the replica's access cost that is affected by replica size and network bandwidth. Both evaluations are combined together, and the replica with minimum value will be replaced by the newly created replica.

2.4.6 Replica Optimization

Replication is a long-term optimization technique which aims at reducing average job execution time in a data grid [118]. The idea of replication is to store copies of data at different locations so that the data can be easily recovered if one copy at one location is lost or made unavailable. Data replication has two direct improvements on the performance of the data grid. One is to speed up data access, which leads to a shorter execution time of grid jobs; and the other one is to save bandwidth between sites, which can avoid network congestion with the sudden frequently required data.

However, replication is also bounded by two factors: the size of storage available at different sites within the data grid and the bandwidth between these sites [88]. Furthermore, the files in a data grid are mostly large [25, 29]; so, replication to every site and hosting unlimited number of replicas would be unfeasible. Therefore deciding the optimal number of replicas and the optimal locations is needed. In this context, the process of how to determine the optimal number of replicas and their optimal locations; is termed as the *replica optimization problem (ROP)*. Replica optimization problem and optimization problems are terms used in the literature alternatively for the same meaning.

ROP could be formulated as follows: given a network topology, the locations of users and their request volumes of a certain popular file, and a set S of sites as potential replica sites, find a subset of S to place number of replicas so that some performance measure is optimized. The goal can be simply to minimize a cost function, such as the cost of accessing the data files by users, or this cost plus some

additional costs, such as the cost of storing replicas, or cost of transferring replicas from replica sites. This cost function is then minimized or maximized (based on what makes sense for the actual cost function) subject to zero or more constraints. For example, such constraints as the maximum storage space allocated for data files on each site (storage constraints), a maximum number of replicas in the system (replica constraints), and number of sites that can host the replicas of data files (site constraints).

The common cost functions that are used in the literature [35, 43-46, 119, 120] are listed below:

Communication Cost (Read Cost): a lot of research studies considered allocating replicas to sites that minimize the read cost [46, 119, 120]. Read cost is usually defined as the cost of transferring a file over the data grid system to the end user.

Storage Cost (Replication Cost): the cost of storing a file at a certain site [35, 43-46]. The storage cost might reflect the size of the file, the throughput of the site, or the fact that a copy of the file is residing at a specific site, which is also called replication cost.

Access Cost: the time taken to access the data files in replica sites [31].

2.4.7 Number of Replicas

The denser the distribution of replicas is, the shorter the distance a client needs to travel in obtaining a copy of the replica [45]. In other words, increasing number of replicas would lead to higher data availability. However, given the size of resources

included within a data grid, the cost of maintaining multiple copies of resources (i.e. data files) and storing them in the data grid system would be expensive; therefore, the number of replicas should be bounded. A mechanism for creating replicas that allows the achievement of availability and performance goals without consuming undue amounts of storage and bandwidth is thus needed.

The work in [43] suggested a algorithm that helps to determine number of replicas needed to maintain the desired availability in P2P communities so that each site within the data grid is authorized to create replicas for the files. The availability of a file depends on the failure rate of peers in the network. A function has been developed to calculate the number of replicas needed for a certain availability threshold. However this algorithm has disadvantages: firstly, the exact number of replicas is not determined; rather it depends on the location service accuracy which depends on the existing number of replicas. The accuracy of the replica location service determines the percentage of accessible files, and thus if the location service is ineffective, more replicas are created to ensure data availability. Secondly, the replica deletion mechanism is not considered, thus the storage cost may be increased.

Meanwhile in [44, 45], the authors had not taken into account the issue of availability to determine the number of replicas. The problem of determining number of replica has been formulated as follows: given the amount of workload a replica server can handle (D), find the minimum number of replica so that the maximum workload is not more than (D).

Furthermore, [46] proposed an algorithm formulated by using a dynamic programming-based algorithm. The purpose of their proposed algorithm is to find the optimal number of data file replica over data grid systems, so that the read cost (transferring file over the data grid system to the end-user) and the cost of storage (site building cost) can be minimized. The drawback of those approaches [44-46] is that storage capacity has been neglected. As a result, if the site has insufficient space, it will not be chosen to host the replica even if it offers low overall cost.

Another variable was investigated by [37] who identified the number of replicas that need to be created, based on the access frequencies of each file that has been requested. By calculating the quotient of average access frequency of popular file divided by average access frequency of all files, the number of replica can be determined.

Meanwhile [35] proposed a replication strategy that makes replication decisions whether to increase the number of replicas to face the high volume of requests, or to reduce the number of replicas to save more storage space. Evidently, increasing the number of replicas will decrease the response time, but the storage cost will be increased accordingly [35].

Table 2. 1: Summary of existing work in determination number of replica

Authors	Technique	Variables	Methodology
Kavitha Ranganathan [43]	Dynamic Placement Algorithm	▪ availability	Determine how many replicas are needed to maintain the desired availability
Yi-Fang Lin et. al [44, 45]	Optimal Number of Replica	▪ workload (sum of data requests [44])	Given the total amount of workload a server can handle, then decide the minimum number of replica
Yaser et. al. [46]	Optimal Number of Replica (ONR)	▪ read cost ▪ storage cost	Find number of replica so that the overall cost is minimized
Ruay-Shiung Chang et. al. [37]	Latest Access Largest Weight (LALW)	▪ access frequency	By calculating the quotient of average access frequency of popular file divided by average access frequency of all files, the number of replica is determined
Husni Al-Mistarihi and Chan Yong [35]	Replica Management in Grid (RmGrid)	▪ replica request ▪ storage cost	By increasing the number of replicas of the most valuable files and decreasing the number of replicas of the less valuable files

2.4.8 Replica Placement

Replica placement is the process of identifying where to place copies of replicated data files within a data grid system. Transferring a data file from a site to a client consumes an amount of bandwidth. One challenge that is raised from this is to locate candidate sites where the replica could be hosted [25] so as to minimize the amount of bandwidth used.

In [22], Rangthan and Foster introduced six replication strategies. They compared those six strategies by measuring average response time and the total bandwidth consumed for each strategy. The lower the response time and the lower the bandwidth consumption, the better the replication strategy is. However, there is a trade-off between response time and bandwidth consumption. The authors concluded that if users are focused on lower response time, then the Cascading strategy would be the best option. On the other hand, if users prefer the consumption of bandwidth to be the most important issue, then Fast Spread is the better choice of all the six strategies. Nevertheless, these two strategies also do not consider storage cost. If a particular file is no longer popular, it will still be stored by the storage element. That will therefore be a waste of free storage. In the Fast Spread replication strategy, the replica is copied to every node it visits when it is brought backward to the requesting node. In contrast to Fast Spread, Modified Fast Spread (MFS) [121] does not necessarily copy the replica to every node it visits when it is brought backward. It is copied to the visited node in two cases. The first case is if the visited node has sufficient free storage space to store the requested replica. The second case is if the node's free storage space is less than the size of the requested replica, and this

replica was found more important than a group of existing replicas that their sizes are greater than or equal to the size still needed to make the node's storage able to store it.

In a different approach, the authors of [40] proposed a dynamic maintenance strategy called Dynamic Maintenance Service (DMS) to improve the performance of the grid environment. DMS decides where to place the replicas based on two main parameters: request frequency and free storage space. However, the replica deletion mechanism is not considered; rather the system does not locate the replica at a site unless there is enough space even if it brings benefit to system performance.

Meanwhile, [39] proposed a replica placement scheme that tries to overcome the bottleneck caused by increasing the downlinks, which are occurring at the same time. The proposed strategy chooses the best site to host the replica according to the evaluation result based on the number of user request and transmission cost. The purpose of the strategy is to replicate the file to a site that provides minimum average transmission cost. Transmission cost is defined to be inversely proportional to bandwidth, and the site that provides the minimum average transmission cost is selected.

Following the bandwidth aspect, [118] proposed a replication strategy, called Bandwidth Hierarchy based Replication (BHR) to reduce access time by avoiding network congestion. BHR reduces the time taken to access and transfer the file. It

places a replica at a high bandwidth location. However, such an approach only considers transmission cost and does not guarantee to minimize the overall cost.

A load balancing replication strategy has been proposed by [41], where the most frequently accessed file is placed closed to the users and the decision of replica placement is made based on the access load and the storage load of the candidate replica servers and their sibling nodes. In relation to this, [122] discussed various replication strategies namely; MinimizeExpectedUtil, MaximizeTimeDiffUtil, MinimizeMaxRisk, and MinimizeMaxAvgRisk while considering the utility and risk indexes, and making the replica placement decision by optimizing the average response time. They concluded that considering both current network state and file requests are better than considering the file requests alone.

Meanwhile, [25] proposed a static replica placement algorithm to place replica files in best p candidate nodes that minimizes the total response of each site by using Lagrangean Relaxation, which is a heuristic approach [123] to measure the response time of each client node to its nearest server node. The algorithm is most likely the p-median problem. They also used the user requests and network latency as parameters to decide when to maintain replica dynamically.

Work by [119] suggested an algorithm that is formulated by using a dynamic programming method to find optimal placement k replicas of an object, such that the overall cost (i.e. storage cost plus read cost) is minimized. Read cost is defined as the data transfer cost and storage cost is the cost of placing replicas at the sites. However, the algorithm does not guarantee the availability of the file, as the priority

choice of location is given to those who provide cheaper services regardless of its availability.

The authors in [44, 124] proposed a placement algorithm so that the workload of user requests among the replicas is balanced. The workload is defined as number of requests that a server satisfies. Given the data usage and maximum workload allowed for each replica server, they suggested algorithm can efficiently determine the minimum number of replicas required. On the other hand, the authors in [43] suggested a algorithm that provides a function that evaluates the placement of replica. The objective of this function is to maximize the difference between the replication benefits and replication cost (storage cost and transfer time). The benefit is the reduction in transfer time to the potential users, the storage cost is the storage cost at the remote site, and the transfer time is the duration from the current location to the new location. Yet again, the replica deletion mechanism was still not considered, thus the storage space cost may be increased.

Then [50] proposed an improvement, namely in the form of the Proportional Share Replication policy. The method places replicas on the optimal locations when the number of sites and total replicas to be distributed is known.

Meanwhile, the work on replication algorithm by [42] had resulted in a Popularity Based Replica Placement (PBRP) algorithm for hierarchical data grids. The idea behind PBRP is to place replicas as close as possible to those clients that frequently request data files. Further work by [106] presented a replica placement strategy in multi-tier data grid that categorized the files based on their access frequency into two groups: 1) Most Frequent Files (MFF) that are replicated and placed at the parent

node of their respective best clients, where the best client for a file is a client which generates the maximum request for that file, and 2) Least Frequent Files (LFF) that are placed at one tier below the root of the data grid along the path of their best client. In [110], a dynamic placement algorithm was proposed that takes into account the dynamicity of sites in the data grid, since a site can at any time leave the grid and possibly join again later. Thus, two parameters were investigated: the request number for each file by each site, and utility of each site that involves the number of times the site did not answer to a file request due to its absence from the grid.

Then, [37] proposed a replication mechanism that replicates the popular file to a suitable site according to the access frequencies for each file that has been requested. Access frequency is an essential parameter that should be taken into account when determining replica placement. However, some important parameters such as overall cost (i.e. storage cost and read cost), distance and availability should not be neglected; otherwise the overall system performance is degraded.

The work presented in [125] suggested a replica placement mechanism that deploys an agent at every site that holds the master copy of files for which replicas are to be created. The main function of each agent is to select the candidate site for placing the replica based on response time and job execution time. The replica is placed at the site that minimizes the time taken for obtaining all the files required by the job. However, storage capacity is ignored as a result if the site is full and provides the minimum response time; it will not be selected to host the replica. On the other hand, a priori replica placement was proposed in [126], where the replicas are created and

placed before starting jobs and launching any work on the grid. The replication is performed at once after the original copies are created and before any file request has been made. The main objectives are to maximize the distance between identical replicas and to minimize the distance between different replicas, so that each site can find the different replicas faster within its vicinity. However, this approach does not cope with the dynamicity of grid environment, and moreover, the storage capacity is not taken into consideration.

Table 2. 2: Summary of existing work in replica placement

Authors	**Technique**	**Variable**	**Methodology**
Chih-Ming Wang et al. [39]	Fair Replica Placement	▪ Bandwidth ▪ Number of user request	Duplicate the file to a node that provides minimum average transmission cost (maximum bandwidth)
Kavitha Ranganathan [43]	Dynamic Placement Algorithm	▪ Storage Cost ▪ Transfer Time	Maximize the deference between replication benefit and replication cost
Mehran Garmehi et al. [119]	Optimal Placement of Replicas	▪ Communication/read Cost ▪ Storage Cost	Place the replica so that the overall cost (read and storage cost) is minimized
Chao-Tung Yang et al. [40]	Dynamic Maintenance Services (DMS)	▪ Request Frequency ▪ Storage Capacity	If the request frequency of file is more than the maximum request rate, and there is free space in the site then DMS

			will duplicate file to that location
Ruay-Shing et al. [37]	Latest Access Largest Weight (LALW)	▪ Access Frequency	According to the access frequencies for each file that has been requested, a popular file is replicated to a suitable site

Through the literature review on this subject, it was concluded that there are many drawbacks of the current replica creation strategies, and there is a need for enhancing these strategies even further. Obviously, in order to get benefits from replication strategies, the storage cost and the read cost must be minimized. From the literature, it was observed that there is a lack of suitable current replication algorithms for the management of data grid resource usage (i.e. network and storage resources), and thus more enhancement is needed.

2.5 Replica Management Stage

At this stage, the replicas have already been distributed to different locations based on the described replica creation algorithms. If multiple replicas exist, a replica management service is required [91]. Replica Management Service (RMS) discovers the available replicas and selects the best replica that matches the user's quality of service requirements, and then adjusts the location of those replicas. Thus, there are two main phases in the Replica Management Stage (RMS); namely Replica Selection (RS) and Replica Maintenance (RM), and each of which will be discussed

briefly in the next subsections since the focus of this thesis is not on this stage of the system, as mentioned previously.

2.5.1 Replica Selection

In a typical grid environment, replication systems create multiple copies for the same data file, and distribute these copies (replicas) to different site locations. These site locations vary in their capabilities, resources, and network. Thus, there is a significant difference in selecting one replica location among many locations that are widely distributed [127]. The replica selection strategy is the process of choosing a replica from among those spreading across the grid sites based on some characteristics [100, 128].

The big challenge of any replica selection strategy is defining the appropriate criteria to determine the best replica location, and the selection algorithm used for replica selection strategy. One common selection criteria would be response time [23] and there are various replica selection approaches discussed in the literature [129-131].

2.5.2 Replica Maintenance

Due to the dynamic nature of data grids, the candidate site that holds replicas may currently not be the best sites to fetch replicas in subsequent periods [91]. Therefore, replica maintenance is needed to relocate replicas to different sites if the performance metric degrades.

In the replica maintenance phase, a replica will be adjusted/moved to the appropriate location based on the information collected relating to some effect factors [34, 40]. Replicas should be adjusted to the appropriate locations that are closer to the computing devices in order to adapt the current network environment to reduce time when the computing device accesses the data, as well as to maintain optimal performance of the network environment.

On the other hand, the network environment is changeable, which makes the same replica sites not always being the best choice to download data while reducing transmission time. Therefore, replicas should be adjusted to the appropriate locations for achieving optimal access times.

2.6 Summary of Chapter

In this chapter, the background on the issues that are covered in this thesis has been provided. This chapter was divided into two main parts; first part presented a brief description and characteristics of data grids and some challenges of the applications running in such environment. The first part concluded that data grids are different from other types of distributed systems due to heavy of computational requirements, wider heterogeneity, higher autonomy of individual entities, and the presence of VOs.

The second part presented the need for a replication algorithm in a data grid environment. Related works of data replication was analytically investigated and presented. This included the related research and progress in replication algorithms,

and recent works in this research domain. The second part concluded that the existing replication algorithms still require more improvement in two aspects; the first aspect is concerned with the algorithm as a whole, where more functions need to be included in the replication algorithm, such as: determining the files to be deleted and how many replicas, and finding the places from where to delete the replicas. To the best of this author's knowledge, there is no any work that has considered all of the core functions together in one algorithm. In this context the functions are:

1. Determining which files to be replicated and how many files;
2. Determining which files to be deleted and how many files;
3. Finding the places to host the newly created replicas;
4. Finding the places from where to delete the replicas; and
5. Determining the file to be replaced in case of insufficient storage space.

The second aspect is concerned with the individual functions of the replication algorithm, where some parameters have been neglected by other replication algorithms, which should be considered. For example, in evaluating the files to determine which file is to be replicated and deleted, the implementer needs to consider the dependency relationships between files, and the decay or growth rate of the file demand. In placing the newly created files, the implementer needs to consider the maximum workload of each site before deciding the placement of replica files.

In the response to the literature survey presented in this thesis, it is proposed that there is a need for an enhanced replication algorithm that embodies all the core functions listed above, and moreover, it should include the neglected parameters by other works as discussed in this chapter. In the next chapter, the methodology and simulation setting employed in this research work will be deliberated upon.

CHAPTER THREE
RESEARCH METHODOLOGY

The purpose of this chapter is to present the approaches used in executing the research. It starts by elaborating on the research phases. Then, each phase is discussed separately in different sections. In Section 3.1 the analysis phase is discussed, followed by the design phase and the main components of the new algorithm in Section 3.2. Section 3.3 presents the implementation phase with a description of the simulator that has been used to implement the new algorithm and how it is integrated in the simulator environment. The evaluation phase and the experimental simulation setup are explained in Section 3.4, while Section 3.5 provides a discussion of how the results of the simulator are verified and validated. Finally, the conclusion of this chapter is presented.

3.1 Introduction

Methodologically computer science researches can be divided into theoretical, experimental and simulation research. However, one research may fall into one or more of these methodological research areas [132].

In data grid environment, the interaction of users and sites are highly complex, which in turn leads to difficulty in predicting their behavior from the first principle.

Therefore, implementing a prototype and deploy it on a testbed with the aim of studying the effects of data grid components (users and sites), will cost time and effort. So, the only way this can be done is by simulation [31]. It is very important to

emulate a grid environment as realistically as possible and provide some methods of evaluating various strategies that might be used to optimize the grid [47]. Hence, Experimental research methodology is the appropriate methodology for such research. Table 3.1 shows the core activities in the methodology used in conducting this research and fulfilling the objectives of thesis.

Table 3.1: Steps of experimental research methodology

	Step	**Output**
1	Analyzing the research problem	**Review of literature:** Criticize the current works and determined the weakness and the gaps.
2	Designing the new algorithm	**Identify current mechanism specification for:** • Conceptual Framework • Proposed algorithm process
3	Implementing the new algorithm	• Refine the simulation code • Finalize the algorithm
4	Evaluating the new algorithm	• Specify simulation environment • Specify performance metrics • Raw data collection • Final result Data analysis and interpret

The core activities are divided into four steps and discussed in detail throughout this chapter.

1. The first step aims to perform an in-depth study on the replication strategies in data grids and surrounding issues. In this phase, the strengths and weaknesses of replication strategies are identified, while areas that should be eliminated and enhanced are explored.

2. The second step works with the design of the new replication algorithm. The design processes involved include determining the design requirements, objectives, specification, and justification for the new algorithm.
3. The third step works with the implementation, validation, and verification of the design of the new replication algorithm.
4. The fourth step executes a performance evaluation of the new algorithm by comparing it with the existing algorithms.

Next sections describe the detail of each step of the adapted research methodology.

3.2 Analyzing the research problem

The first step of the work is the analysis of the problem. This step was commenced by dividing the problem into three sub-problem domains: 1) determination of files that have to be replicated or deleted, 2) identification of number of copies, and 3) determination of where to place the replicas. Having divided the problem into three parts, the main components that are involved in each of the sub-problem are identified and analyzed.

Once there is a clear understanding and a good idea of what was the problem and the factors that affect the problem, the second step in this phase involved exploring existing works and knowing how other researchers tackled the problem and what the identified research challenges were. In this step the current works were criticized and the weaknesses were determined in order to illustrate and frame the gaps. A detailed outcome of this process has been presented in Chapter two.

3.3 Designing the new algorithm

In this step, the core strategies of the proposed algorithm have been developed. The development of these strategies includes the development of basic mathematical models of each strategy in the proposed algorithm.

The components of the new algorithm that is termed as Data Replica Creation Algorithm (DRCM) are shown in Figure 3.1, where DRCM includes three main strategies namely, 1) Replica Creation/Deletion Strategy (RCDS), which decides on replicas creation or deletion of files and how many replicas; 2) Replica Placement Strategy (RPS), which finds the best location sites to host the newly created replicas, and the best site from where to delete replicas, and 3) Replica Replacement Strategy (RRS) that determines a victim replica to be deleted in order to make room for a more critical replica, in case of insufficient storage capacity at the target storage element. A full description of DRCM components and the detailed process of the involved strategies are explained in Chapter four.

The proposed algorithm has been designed with the aim of considering both users' and system's perspectives. Therefore, each strategy of the algorithm contains one or more components that reflect both users' and system's perspective. For example the components that reflect users' perspective are: Access history, Dependency level, Read cost, and File value. However, the components that reflect system's perspective are: Number of existing replicas, Locations of replication sites, Workload of each site, Bandwidth, Free space, and Size of replica.

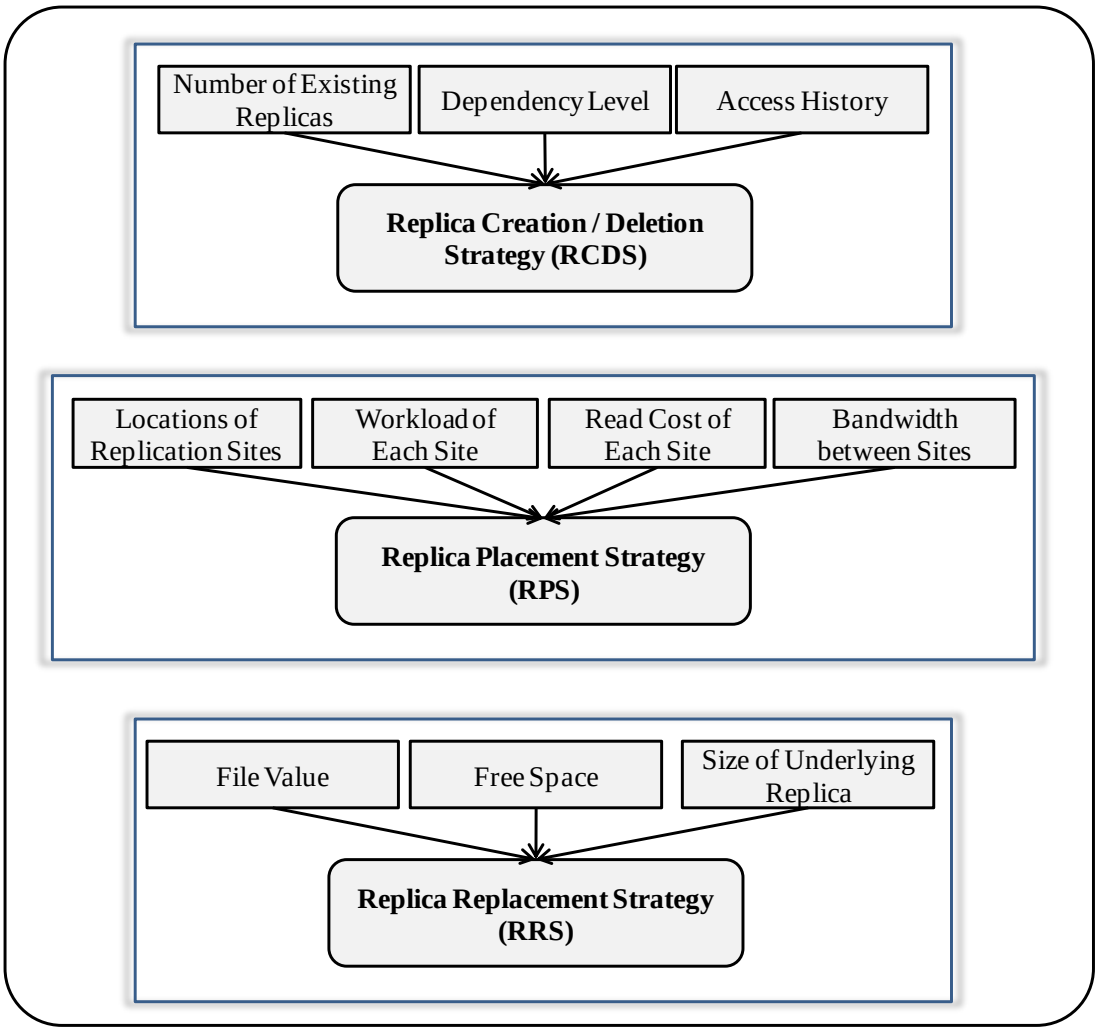


Figure 3.1: Abstract view DRCM algorithm

DRCM has achieved a set of functional and non-functional requirements. The non-functional requirements that have been achieved include:

Scalability: The DRCM has the ability to scale well, in case of increasing the depending parameters, such as a size of files, number of submitted jobs, etc.

***Interoperability*:** The DRCM should have the ability to work with other existing systems without special effort, such as Replica Location Services (RLS).

Performance: The DRCM should outperform other similar algorithms. Therefore, the evaluation metrics are used to measure the performance of the system.

Feasibility and Simplicity: The DRCM must be feasible to implement, thus the DRCM has been kept as simple as possible.

The functional requirements of DRCM include replica creation/deletion decision, replica placement, and replica replacement. In the replica creation/deletion functionality, DRCM decides whether to perform replication/deletion processes or not, and if so it determines which file will be replicated/deleted and how many copies to be created/deleted. In order to make the decision, DRCM depends on the popularity of the files combined with its dependency level which is termed as File value (FV), and the existing number of replicas of each file. The DRCM deploys exponential growth/decay approach combined with file dependency to evaluate the files, which is to determine the files' power in terms of users' perspective. On the other hand, the DRCM evaluates the files based on its value and existing number of replicas with the aim of determining the files' power in terms of system's perspective. Then, based on the two perspectives, the DRCM makes the decision.

Relating to the replica placement functionality, in order to host the newly created replicas, the location sites that provide the least amount of Read Cost (RC) are

chosen. Choosing the location sites depends on five parameters; 1) the bandwidth among the sites, 2) the RC, 3) the location of replication sites, 4) the location of dependence files, and 5) the workload of the sites.

Eventually, in replica replacement functionality, the deletion function is invoked to delete a victim replica from the target storage element in order to provide room for the newly created replica. Choosing the victim replica depends on two parameters, namely the value of the files and the size of the file.

The DRCM works in the background of the system in such a way that there is no direct connection with users. DRCM relies on other existing data grid core services, such as Replica Location Services (RLS) [22] that provides information related to the physical file locations, and Information Service Provider (ISP) [100] such as Network Weather Services [133] to provide the network availability and status.

Therefore, as shown in Figure 3.2, DRCM offers the following functionalities:

1. gathers replica locations information from RLS;
2. gathers network bandwidth information from the NWS;
3. gathers jobs information from the history file; and
4. makes main decisions namely, replica creation, replica placement, and replica replacement.

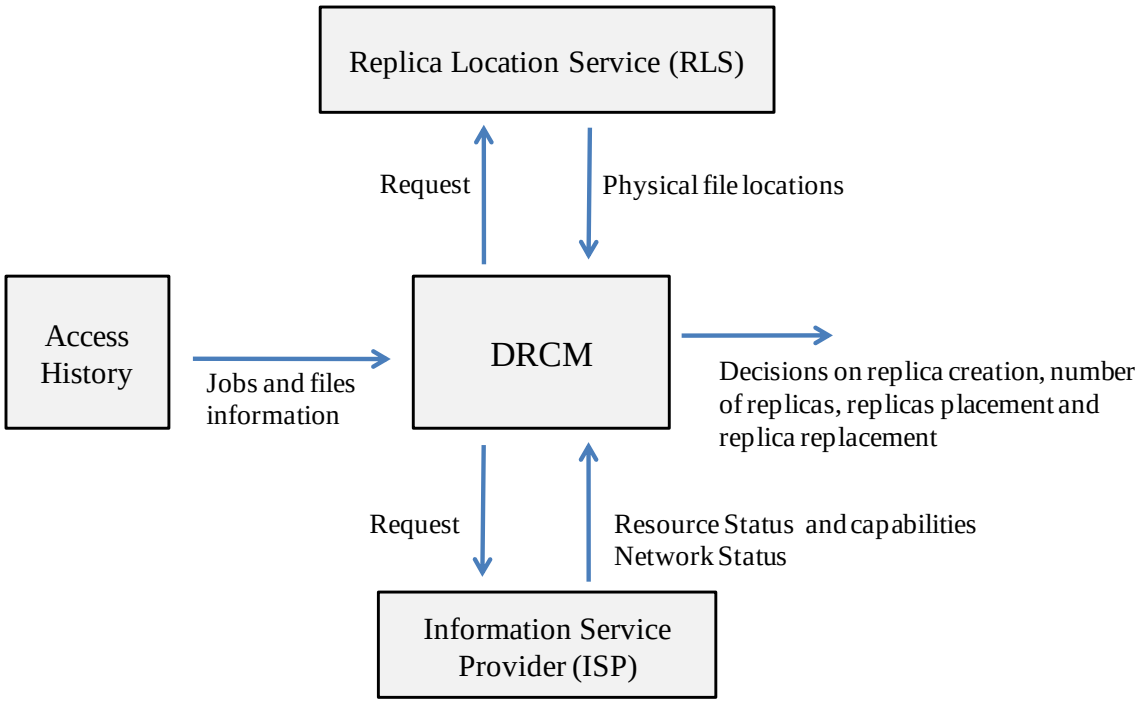


Figure 3. 2: Overview of DRCM and other related entities

3.3.1 Comparison of DRCM algorithm with other existing algorithms

DRCM algorithm as a part of data management services in data grid provides the grid community with one of the essential services. The idea of DRCM inspired from the real world marketing community, where there is a need for balance between the interests of users and the interests of service providers (resources). A combination of three strategies are embodied in one algorithm, the key advantage of DRCM is that it takes into consideration both users' and system's perspectives. Table 3.2 shows a comparison between five replication algorithms based on the three strategies. Table 3.3 shows a comparison between five replication algorithms based on the existed features.

Table 3. 2: Comparison of replication algorithms based on included strategies

Algorithm	Replica creation Decision	Replica placement	Replica replacement
LRU	Always replicate	Based on NoA	Least recently used
LFU	Always replicate	Based on NoA	Least frequently used
Economy	Prediction function	Based on NoA	Prediction function
LALW	Replicate with condition	Based on NoA	Least frequently used
DRCM	Replicate with condition	Based on NoA and the current situation of sites	Based on file value and storage cost

Table 3. 3: Comparison of features exist on DRCM algorithm and other replication algorithms

	Feature	LFU, LRU, Economy	LALW	DRCM
1	*Number of files to be replicated*	One	One	Depends on file value and number of existing replicas
2	*Number of created replicas*	One	Depends on NoA	Depends on file value and number of existing replicas
3	*Including deletion function*	No	No	Yes
4	*Replica placement*	Places replica at requested site	Depends on NoA	Depends on read cost, workload, replication sites, and places of dependence files
5	*Types of files*	Independent files	Independent files	Independent and dependent files

One of the uniqueness of DRCM over other replication algorithm is that it decides to replicate more than valuable files if it is necessary contrary to existing algorithms that replicate only one file in one decision. DRCM adjust the availability of the files by increasing or decreasing appropriate number of replicas based on file value and existing number of replicas. Moreover, another key advantage of DRCM is that it introduces a deletion function that removes unwanted replicas from the system. Thus, increase the storage space availability. Increasing storage space availability leads to decrease number of times the system invoke replacement strategy and reduce the processing time and improve the performance of other replication strategies. In addition, existing works operate only on independent files, contrary to DRCM that introduces dependency level, which makes it applicable for both dependent and independent files.

Figures 3.3, 3.4, 3.5, and 3.6 illustrate the steps of DRCM, LALW, LRU, and Economy algorithms respectively, which shows the difference of the procedures of each algorithm.

The key advantage of DRCM is that it takes into consideration both users' and system's perspectives in file evaluation and determining the popularity of files. Between line 2 and 7 of the algorithm, as shown in Figure 3.3, DRCM identifies the File Power for users which is denoted by FP_{user} and File Power for system that is denoted by FP_{system} . Based on FP_{user} , FP_{system} , and Threshold value (TH, that will be discussed in Chapter 4), the Estimated Number of Replicas (ENoR) is calculated as shown in line 9. Then, based on ENoR value the files are categorized into three groups as shown between lines 10 to 15. Comparing with LALW algorithm that is

shown in Figure 3.4 (lines between 2 and 10) file evaluation totally depends on number of access that reflects users' perspective only. It is worth mentioning that computation time of DRCM in this phase more than computation time of LALW, as it determines the group of files to be replicated or deleted. Nevertheless, DRCM accelerates the process of placing replicas. DRCM rarely invoke replacement strategy, this is because it deletes all of the unwanted replicas in replication decision phase before deciding where to place the replicas. Contrary to LALW that has no deletion function, it depends on replica replacement strategy to delete the replicas in order to make a free space for the newly created replicas. Lines between 21 and 27 in Figure 3.4 shows the steps of deleting replicas in case of insufficient storage space in LALW algorithm. In line 26 there is *While Loop* means that deleting replicas will be continuing until there is a free space. Contrary to DRCM that deletes only one file, the steps shown between lines 42 and 49.

DRCM algorithm:

Input: Number of Access of each file ($NoA(file_i)$), Number of file intervals t, Dependency Level (Dl), File Size, Bandwidth between sites, Number of existing copies of each file ($NoC(file_i)$);

Output: Duplicating a certain number of files that should be replicated to appropriate sites

Procedures:

1: ***/* Replica Creation/Deletion Strategy */***
2: **for each** files in the grid
3: Calculate $r \leftarrow \sum_0^{t-1} r_i/t-1$
4: Calculate $FL \leftarrow N_f^t \times (1+r)$ /* N_f^t number of access in the last time interval */
5: Calculate $FW \leftarrow \sum_{i=1}^{n} FL_i \times DL_i$
6: Calculate $FV(t,f) \leftarrow FL(t,f) + FW(t,f)$
7: Calculate $FP_{users} \leftarrow \frac{FV}{\sum_{\forall files} FV}$ /* equation 4.3
8: Calculate $FP_{system} \leftarrow \frac{NoC}{\sum_{\forall files} NoC}$ /* equation 4.4
9: Calculate ENoR $\leftarrow \frac{(\mathrm{FP}_{\mathrm{users}} - (\mathrm{TH} \times \mathrm{FP}_{\mathrm{system}})) \times \sum_{\forall files} NoC}{\mathrm{TH}}$ /* equation 4.7
10: **if** ($ENoR > 0$) then
11: Add $file_i$ to *Popular_List*
12: **else if (**$ENoR < 0$) then
13: Add $file_i$ to *Unwanted_List*

14: **else if (**$ENoR = 0$) then
15: Add $file_i$ to *Stable_List*
16:
17: ***/* Replica Pplacement Strategy */***
18: **for each** file in *Unwanted_List*
19: List all sites that contain $file_i$
20: **for each** site in list
21: Calculate $FV_{s_i} \leftarrow \frac{File\ Value}{NOR_{s_i}}$ /* equation 4.9
22: Calculate $FTT \leftarrow \frac{File\ Size}{Bandwidt\ h}$ /* equation 4.10
23: Calculate $RC \leftarrow \frac{\sum_1^n FV_{s_i} \times FTT}{m}$ /* equation 4.8
24: DescendSort [$site_j$]
25: **while** ($ENoR(file_i) \neq 0$)
26: Delete $file_i$ from $site_j$
27: $ENoR(file_i) + +$
28: Break;
29:
30: **for each** file in *Popular_List*
31: **for each** site in grid
32: **if (**$site_j$ fulfill the requirement for hosting $file_i$) then
33: Calculate $FV_{s_i} \leftarrow \frac{File\ Value}{NOR_{s_i}}$ /* equation 4.9
34: Calculate $FTT \leftarrow \frac{File\ Size}{Bandwidt\ h}$ /* equation 4.10
35: Calculate $RC \leftarrow \frac{\sum_1^n FV_{s_i} \times FTT}{m}$ /* equation 4.8
36: add $site_j$ to *candidate_site_list*
37: AscendSort [$site_j, file_i$]
38: **for each** site in list
39: **while** ($ENoR(file_i) \neq 0$)
40: add ($site_j$, $file_i$) to *best_site_list*
41:
42: ***/* Replica Replacement Strategy */***
43: **for each** file in *best_site_list*
44: check sufficient space in the targeted site
45: **if** *storage_space*($site_j$) $< size(file_i)$ then
46: Calculate $RS \leftarrow File\ Size - Free\ Space$ /* equation 4.13
47: Select files that their size $\geq RS$
48: Sort files in descending order based on their values
49: Delete file that has the smallest value
50: replicate ($file_i$, $site_j$)
51: **End**

Figure 3. 3: DRCM Algorithm

LALW algorithm:

Input: Number of Access of each file ($NoA(file_i)$), Number of file intervals t,

Output: Duplicating a certain number of files that should be replicated to appropriate sites

Procedures:

/* Replica Creation Strategy */
for each files in the grid
Calculate $FW(file_i) \leftarrow 2^{-(t-1)}$
Calculate $AF(file_i) \leftarrow FW(file_i) \times N_f^t$ /* a_f^t number of access of $file_i$ at time t */
Sort files in descending order based on *AF*
Check the file that got maximum *AF*
    if ($AF(file_i) >= \sum AF(files)$) then
        $file_i$ is the popular file
    $AF_{avg}(file_i) \leftarrow \frac{AF(file_i)}{t}$ /* average number of accesses for popular file in a time interval
    Calculate $AF_{avg}(all\ files) \leftarrow \frac{\sum AF(file_i)}{t \times Number\ of\ files}$ /*average number of accesses for all file in a time interval */
    if $AF_{avg}(file_i) > AF_{avg}(file_i)$ then
    $Num_{\boldsymbol{system(p)}} = \frac{AF_{avg}(file_i)}{AF_{avg}(all\ files)}$
    there are ($Num_{\boldsymbol{system(p)}} - 1$) replicas to be replicated

/* Replica Placement Strategy */
for each sites in the grid
Sort files in descending order based on AF
Select the top ($Num_{\boldsymbol{system(p)}} - 1$) sites that got highest AF
Add selected sites in *best_site_list*

/* Replica Replacement Strategy */
for each file in *best_site_list*
    check sufficient space in the targeted site
        if storage_space($site_j$) $< size(file_i)$ **then**
        Sort files in descending order based on their NoA
          while storage_space ($site_j$) $< size(file_i)$ **do {**
          Delete file that has the smallest NoA
          storage_space($site_j$) $= storage\_space(site_j) + size(file_i)$
          }
    Replicate (popular_file, $site_j$)
    End

Figure 3. 4: LALW Algorithm

LRU algorithm:

Input: Number of Access of each file ($NoA(file_i)$),

Output: Duplicating a certain number of files that should be replicated to appropriate sites

Procedures:

```
/* Replica Creation/Deletion Strategy */
    for each files in the grid
        if (file_i requested) then
            List requested sites of file_i
                for each requested site
                    if storage_space(site_j) < size(file_i) then
                    Sort files in descending order based on their NoA
                        while storage_space (site_j) < size(file_i) do {
                        Delete file that has the smallest NoA
                        storage_space(site_j) = storage_space(site_j) + size(file_i)
                        }
                Replicate (file_i , site_j)
    End
```

Figure 3. 5: LRU Algorithm

Input: Number of Access of each file ($NoA(file_i)$);

Output: Duplicating a certain number of files that should be replicated to appropriate sites

Procedures:

```
/* Replica Creation/Deletion Strategy */
    for each files in the grid
    evaluate files based on binomial distribution function
    Sort files in descending order based on file value
    List sites that request file_i
        if (file_i requested) then
            List requested sites of file_i
            Sort sites in descending order based on NoA
            Check storage_space of site that issued maximum NoA
                if storage_space(site_j) < size(file_i) then
                    Sort files in descending order based on their file value
                        while storage_space (site_j) < size(file_i) do {
                        Delete file that has the smallest file value
                        storage_space(site_j) = storage_space(site_j) + size(file_i)
                        }
            Replicate (file_i , site_j)
End
```

Figure 3. 6: Economy Algorithm

3.4 Implementation Phase

Within the data grid environment, the interaction of users and sites are highly complex [134], which in turn leads to difficulty in predicting their behavior. Therefore, implementing a prototype and deploying it on a testbed with the aim of studying the effects of data grid components (users and sites) will cost time and effort. So, as an alternative, simulation can be used [31]. It is very important to emulate a grid environment as realistically as possible and provide some methods of evaluating various strategies that might be used to optimize the grid [47].

There are many simulators that are available to model a grid environment and support grid features, for example MONARC [135], which is very specific to LHC high energy physics but does not provide flexibility in the architecture of the grid and types of grid jobs that could be simulated. GridSim [136], ChicSim [137], and Bricks Grid [138] concentrate on complex scheduling decisions and does not deal with data movement around the simulated grid. The important components for data management simulator [139, 140] should include: jobs or application to run on the grid, computing resources to execute the job, storage resources to store data, network infrastructure to connect the resources, a scheduler to decide where jobs should be sent, a transfer system to move data around the grid, and a component to perform replica management. The simulation should also support grid features in any particular replication management aspect, and should be able to accept different grid topologies and workloads. Furthermore, it also should be able to investigate different scheduling and replication strategies and yield enough information to evaluate the performance of these strategies.

3.4.1 OptorSim

In this thesis, OptorSim [107, 141, 142]simulator was utilized to simulate the proposed replication algorithm. The main idea of OptorSim is when given a grid topology, resources, and a set of jobs and optimization strategy, it can simulate data movement around these job runs and supply information on various factors that could be used to evaluate the performance of the optimization strategy. The key advantage of OptorSim is that it is much closer to reality since it is based on the EU DataGrid architecture [107], which is why it is widely used by researchers of various replication strategies in conjunction with different job scheduling algorithms in data grid environments [47, 142, 143].

OptorSim is capable of providing a testbed similar to the original data grid environment by providing multiple grid sites with storage elements that can be used to create and store replicas. Users can also set the parameters of OptorSim according to their requirements to run the simulation. In the following subsection, the architecture of OptorSim is explained. A more detailed architecture and implementation of OptorSim can be found in [37, 42, 48, 144-146].

3.5 Evaluation Phase

Upon completing the implementation phase, the experimental set-up (choosing the appropriate experiment topology and parameters) is performed in the evaluation phase. In order to evaluate system performance, the proposed simulation algorithm output needs to be measured, analyzed, and compared with other algorithms.

3.5.1 Experimental Setup

The study of DRCM was carried out using two models; EU DataGrid [141] sites and LALW [143] sites, and their associated network geometry. In EU DataGrid, a set of high energy physics analysis jobs was generated from the Compact Muon Solenoid (CMS) [37] experiments in the European Organization for Nuclear Research (CERN) [73, 74] project. Jobs were based on the CDF use-case as described in [69].

There were six job types with no overlap between the set of files each job would access. The aim of using two different topologies is to examine the proposed replication algorithm in different environments and evaluate its performance against other existing algorithms. Using different testing environments has the following advantages: firstly, the way the underlying topology affects the behavior of the replication strategies can be investigated. This includes not only the basic structure but also the importance of interconnections between sites, the location of the data source, and how close other sites are to the data source. Secondly, the effects of a grid's scale and complexity can be observed as the number of grid sites is increased.

The EU DataGrid topology includes 20 sites in USA and Europe as shown in Figure 3.3. Within this model, each site, excluding CERN and FNAL, was assigned with a Computing and Storage Element. CERN and FNAL were allocated with Storage Elements only, since they produce the original files and store them.

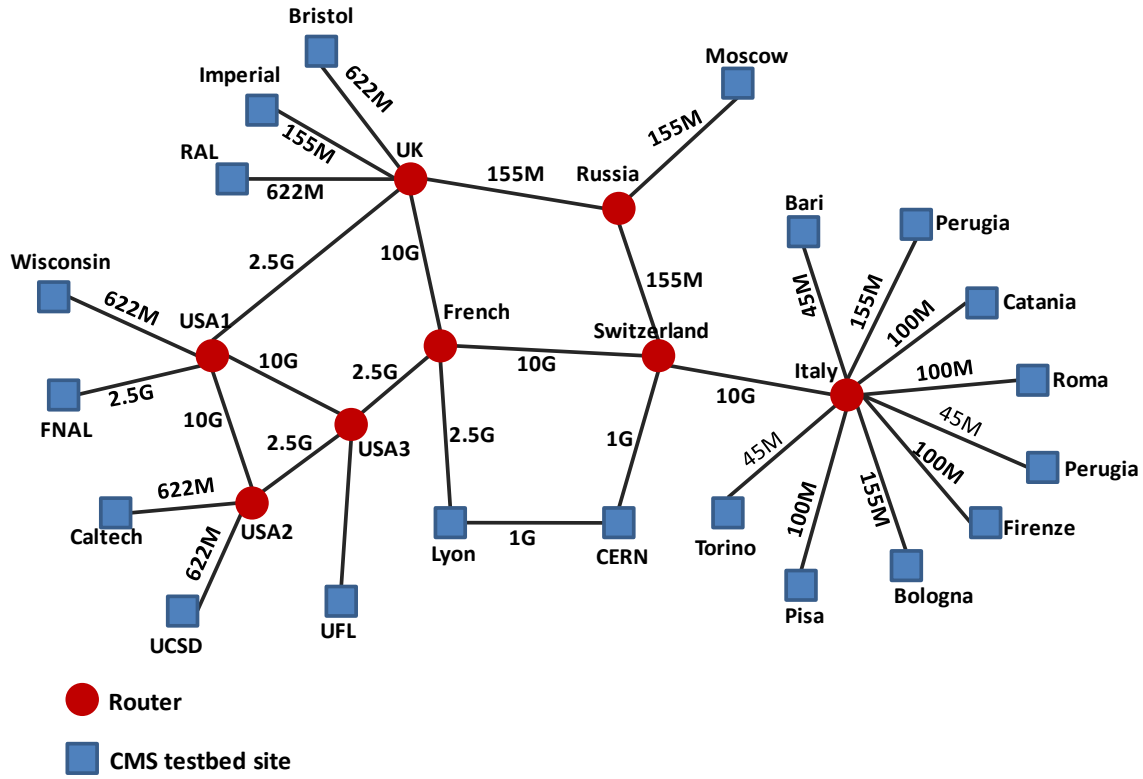


Figure 3. 7: The EU data grid Testbed Sites and their associated network geometry [147]

The LALW [37] testbed has the network topology and resources shown in Figure 3.4. It comprises of 12 grid sites, each site has a storage capacity of 50 GB, while Site8 has 100 GB to hold all the original files. Details of this topology will be discussed in Chapter 5.

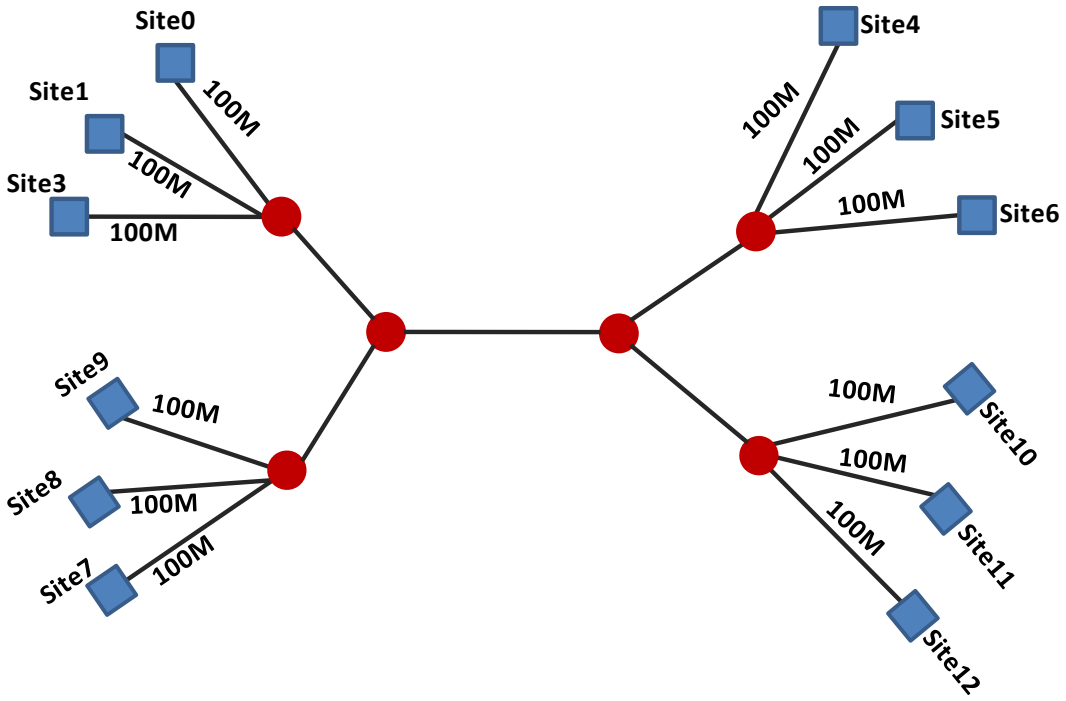


Figure 3. 8: LALW Testbed Sites and their associated network geometry

3.5.2 Simulation Scenarios

DRCM is compared with other existing algorithms discussed in Chapter 2. In order to expose the throughput and system performance, the simulation was run according to different scenarios, and the results are compared with the other existing algorithms. The parameters that influence replication strategies are [37]: number of submitted jobs, site policy, Scheduling Algorithms (job scheduler), Access History Length, Storage Metric (D), Maximum Queue Size, and Job Delay.

i) Number of submitted jobs (job workload)

System scalability can be tested by the number of jobs running during the simulation. According to [49], the estimated number of submitted jobs by users in CMS 2002 experiments was 1000 jobs. So to simulate different number of jobs, the maximum number of submitted jobs was increased by a factor of four and the minimum was decreased by a factor of four, i.e. number of jobs that is considered in our evaluation varied between 200 and 4000 jobs.

ii) Site Policy

Different grid sites are likely to prioritize different kinds of job; each job has its own requirements, which means that there are sites that may not be able to execute certain jobs. Site policy refers to type of jobs which would not be accepted by sites in the system. The effect of site policies on the overall running of the grid is investigated. This was done by defining two extremes of policy namely "All Job Types" and "One Job Type" [31, 47]. Therefore, if a site accepts all job types, then the site has "All Job Types" policy in the underlying grid. Likewise, if a site would accept only one job type, then the site has "One Job Type" policy.

iii) Scheduling Algorithms (job scheduler)

Typically, a scheduler submits jobs to the grid sites according to some algorithms that may affect the performance of the replication algorithm [31]. The following are the job scheduler algorithms implemented in OptorSim:

a) *Random*: jobs are randomly scheduled to any Computing Element that can execute the job.

b) Queue Length (QL): jobs are scheduled to the Computing Element that has the shortest queue of waiting jobs.

c) Access Cost (AC): jobs are scheduled to the Computing Element with the lowest access cost (time taken to access the files required by the job).

d) Queue Access Cost (QAC): jobs are scheduled to the Computing Element with the lowest queue access cost (sum of access cost for all jobs in the queue at the given computing element).

Since the fourth scheduler algorithm combines the second and the third algorithms, this research employed the fourth scheduler and the first scheduler in all the tested scenarios.

iv) Access History Length

This is defined as the period of time for which to keep file access history. The history of file accesses is used by replication algorithms to identify the most popular file in the next time window. Therefore the length of history used in the calculations must be carefully chosen to produce accurate prediction. If the history does not go back in time far enough, the statistics of file access may not be accurate, but if the history goes back too far, it may affect simulator performance. Moreover, the information could be overdue and useless. The ideal length is short enough that is not too time consuming to process and changes in file request patterns are noticeable, and long enough to have a good view of the overall access patterns. Thus the length of access history considered for evaluation varies between 10^3 seconds and 10^6 seconds, where the reasons for which are detailed out in Chapter 5.

v) Storage Metric (D)

It is defined as the ratio of the Storage Element size to the total dataset size [20, 36]

$$D = \frac{Storage\ \ Element\ \ Size}{Total\ \ dataset\ \ Size} \quad (3.1)$$

If the value of $D > 1$, then there is enough space in the storage element to hold all files that a job would require. Hence, there is no need for any deletion and the replication strategy will have little effect on the performance of the grid. If $D < 1$, than the storage element is not capable of holding all of the required files so deletion must take place and choices have to be made on which replicas to keep. In this case the replication strategy will be useful. In order to study the effect of storage metric, different file sizes that vary between 200 to 2000 MB were considered and used in the experiments.

vi) Maximum Queue Size

This is defined as the maximum number of jobs a site can keep in its queue. The queue size was fixed at 200 jobs in all of the experiments, since it is the default setting and used in all existing works [31, 47]

vii) Job Delay

This is defined as the rate at which jobs are submitted to the data grid. The job delay was fixed at 25 seconds in all of the experiments.

3.5.3 Performance Evaluation Metrics

There are a number of performance metrics that can be used to evaluate system performance. The proposed system performance will be tested using the following metrics:

I. **Mean Job Execution Time**: This is defined as the average time a job takes to execute, from the moment it is scheduled to Computing Element, to the moment when it has finished processing all of the required files. It is calculated by summing the total time taken by each job and divided by the total number of jobs [37, 108, 110, 141, 142, 148], as shown in the following formula:

$$\text{MJET} = \frac{\sum \text{T}_{\text{Departure}} - \text{T}_{\text{Arrive}}}{\text{n}} \tag{3.2}$$

where,

T_{Arrive} : start time of job execution,

$T_{Departure}$: completion time of job execution, and

n: total number of processed jobs in the simulation.

II. **Efficient Network Usage**: ENU is defined as a measure of how well the replication strategy uses the network [142]. It is computed as:

$$ENU = \frac{N_{\text{remote file access}} + N_{replications}}{N_{\text{remote file access}} + N_{local\ file\ access}} \tag{3.3}$$

where $N_{\text{remote file access}}$ is the number of accesses that Computing Element reads a file from a remote site, $N_{replications}$ is the total number of file replication that occurs, and $(N_{\text{remote file access}} + N_{local\ file\ access})$ is the number of times that Computing Element reads a file from a remote site or

reads a file locally. A lower value would indicate that the utilization of network bandwidth is more efficient. In order to get low ENU value, the value of (numerator) $N_{replications}$ should be small, which means that the algorithm should not do more replication.

III. **Storage Element Usage:** Storage element usage of a site is the percentage of capacity reserved by files according to the total capacity for the underlying storage. The average of all storage reserve capacity in the data grid can reflect the total system storage cost [141, 142]. The Average Storage Usage (ASU) metric is computed by the following equation [142]:

$$\text{ASU} = \frac{\sum_{i=1}^{n} \frac{U}{C}(\text{site}_i)}{N} \times 100\% \qquad (3.4)$$

where,

U: storage usage space that is reserved by the data files,

N: number of sites in the data grid, and

C: total capacity of the storage medium.

- **Computing Element Usage (CE Usage):** This is defined as the percentage of time that a CE is active (transferring or processing data) during the simulation. The CE usage of the whole grid is computed by aggregating the CE usage of each individual CE. CE usage is a metric that could be of interest to resource owners, as high CE usage would mean that the workload is balanced across the grid [142]. Low CE usage, on the other hand, would mean that some CEs have long queues while others are underused.

3.6 Validation of OptorSim

In order to ensure that the simulation model and its results are correct, validation and verification processes need to be performed. There are some techniques to validate and verify the simulator as presented in [31, 47], which is a standard reference for simulation modeling:

Face Validity: the role of face validity is to validate/approve the conceptual model and architecture of the simulation that had already been presented; in this research, this would be the acceptance of the OptorSim model by experts at conferences and in peer-reviewed journals [149]. Since OptorSim is based on the EDG architecture, as well as the fact that the model was developed by the developers of the EDG data management services, the criterion of face validity would be fulfilled.

The Fixed Value tests were administered via a suite of functional tests, which also verified the correctness of the implementation during development. The functional tests covered basic network behavior, schedulers, access patterns, and replication times and optimization algorithms. These used very simple grid configurations with only a few sites, where the correct results could be calculated and used to check the simulation results in order to show that the implementation was indeed correct. Some simple configurations were also tested in [2], which also showed that OptorSim behaves as expected.

Internal Validity

For the Internal Validity tests, the parameters of the simulation were chosen to give as little randomness as possible (such as in the scheduling of jobs to different sites, or the jobs which were run). OptorSim was then run with the same parameter set 200 times, using the same CPU type each time and using a fixed random seed to give internal consistency. The mean job time was calculated each time and the distribution of times was plotted, as shown in Figure 3.9. Although at first glance this looks like there is a large variance with an RMS value of 55.95, when compared to the absolute values (which are of order 104) it is clear that the variation is in fact only about 0.5%. What variation does exist is due to the fact that in certain situations, when there is a choice between two equally valid options, one will be chosen at random. If the Replica Optimizer at a site, for example, wants to get a certain file in the shortest time possible, and there are two sites which could deliver the file in exactly the same time, one of those sites will be chosen at random. Rather than being due to the random seed for the simulation changing, this is due to the way certain elements, such as hash tables, are implemented in Java. This is therefore an irreducible variation, but it is small enough such that we can confidently state that the simulation is internally valid at the required level.

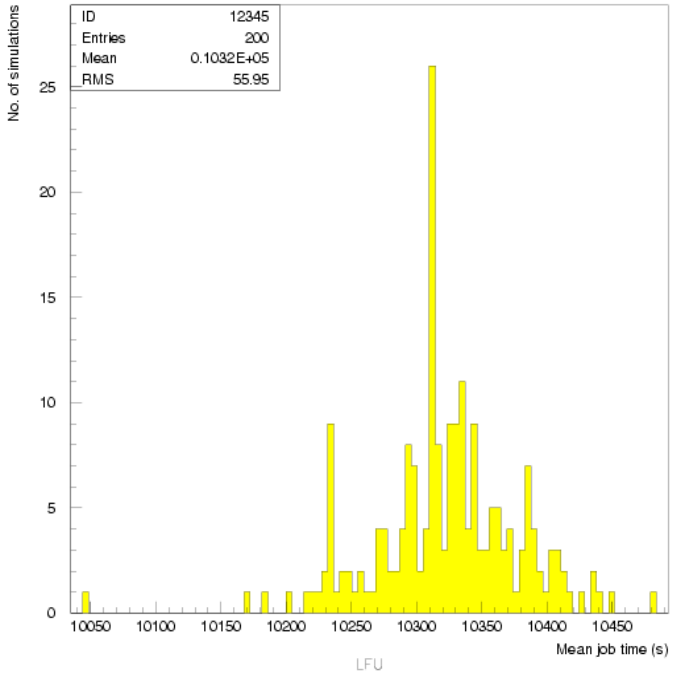


Figure 3.9: Variation in mean job time over 200 simulation runs [107, 108].

Machine Independence

OptorSim was run on a number of different Linux machines at Glasgow and CERN. This resulted in a standard error due to machine type of the order of 1%, which is less than the random variation (both internal and that due to choice of parameters). It can therefore be safely stated that the code is machine-independent in terms of results output.

3.7 Summary of Chapter

This chapter described the approach used in ensuring that the research objectives are fulfilled, verified, and validated. Four main research steps had been explored in this chapter namely, analyze phase, design phase, implementation phase, and evaluation phase.

In the design phase, the main components that should be considered in the proposed algorithm were presented. In the implementation phase, the simulator that was used to test the proposed algorithm was described in greater detail. In this phase, a discussion of the integration of the proposed algorithm in the simulation environment was elaborated upon. In the evaluation phase the three main steps were describes, which are:1) the experimented setup and the topology that will be used in the evaluation, 2) simulation scenarios, and 3) evaluation metrics.

In the next chapter, the simulation experiment is conducted with different scenarios that have been discussed in this chapter, and the results of the proposed algorithm is compared with other similar algorithms

CHAPTER FOUR
STRATEGIES IN REPLICA CREATION ALGORITHM

In this chapter, the requirements and goals of the proposed work is discussed. The implementation of Data Replica Creation Algorithm (DRCM) to improve performance of a grid system is described. To that end, the detailed design of DRCM is presented; and design justifications, specifications, and objectives for the new algorithm are provided. Then, the chapter rounds-off by presenting the details of the proposed mechanisms and its processes.

4.1 Design Goal

The ultimate goal for this research is to create a replica creation algorithm that has properties as described in the following subsections.

4.1.1 Minimizing network bandwidth consumption

The term bandwidth relates to the propagation characteristics of communication systems between two nodes within the network, and bandwidth quantifies the data rate that a network path can transfer [31]. When the network bandwidth consumption is reduced, then network traffic will be decreased. Network traffic can be expressed as transmission time in the network, which is the period of time when a user requests a replica until the replica download is completed. Therefore, the proposed algorithm utilizes bandwidth as optimally as possible by:

a) placing the replicas very close to the sites that request the replicas;
b) distributing the replicas among sites such that the workload is balanced, and thus avoid the network congestion;

c) ensuring that a site only stores a copy of a replica with the aim to distribute the workload among the sites; and
d) the number of times the system performs the replication process is minimized as much as possible.

4.1.2 Minimizing Storage Cost

The storage cost is the storage space used to store data. In this context, both storage cost and storage usage terms have the same meaning and used alternatively throughout this thesis. Therein, the proposed algorithm utilizes bandwidth as optimally as possible by:

a) determining number of replicas that need to be created or deleted such that there is a balance between users' perspective and system's perspective; and
b) determining the victim replica that needs to be replaced by the newly created replica, in case there is not enough space at the chosen site to host the new replicas.

4.2 DRCM Detailed Design

4.2.1 Replica Creation/Deletion Strategy

In a data grid, when a file is required by a job and is not available on a local storage, it may either be replicated or read remotely. If a file is replicated, the next time it is requested, the job can read it quickly and the time to complete the job will be reduced. However, if replicating a file requires the deletion of other files, future jobs

running those files which were deleted will take longer. Therefore, a decision must be made whereby only the most popular files are replicated and the least popular files are deleted. The replication decision combines two main decisions: 1) which file should be created/deleted and 2) how many copies to be created/deleted.

One solution is to evaluate on a file-by-file basis whether a file is worth having compared to all other files that are already there. This requires a strategy to predict the value of each file, based on file access history. The proposed algorithm (DRCM) includes both the users' and system's perspectives. In the next subsections, details of how the proposed algorithm (DRCM) makes the two decisions are presented.

4.2.2 File Evaluation

Due to the limited storage capacity, replication decision should be made to conform to users' needs so that high demand files (replicas) should be kept and low demand files (replicas) are deleted.

Replica creation strategies intend to identify potential popular files because it is believed that popular files in past time window will remain popular in the future time window [150]. Our strategy is designed by tracking two kinds of behavior: 1) **user's behavior** of requesting a file and noting the change to this request whether is a growth or decay change, and 2) **files' behavior** of requesting other file and noting the level of dependency on a certain file.

Tracking Users' Behavior of Requesting a File (File Life Time)

Popularity of a file depends on the number of accesses made to the file by users [24]. With this, popular data files can be identified by analyzing the file access history. Many real world phenomena can be modeled by functions that describe how things grow or decay as time passes. Examples of such phenomena include the studies of populations and bacteria [22, 43, 151]. This work proposes to adopt the exponential growth/decay model in determining popularity of a file. This is due to the fact that each file has its own number of access and this value increases by the increase of access rate and vice versa. If the access rate increases, so does the growth/decay rate.

If we use N_f^t to represent the number of accesses for file f at time t, and N_f^{t+1} to represent the number of accesses at time $\mathrm{t}+1$, the exponential growth/decay model would be given by:

$$N_f^{t+1} = N_f^t \times (1 + r) \tag{4.1}$$

where r is the growth or decay rate in number of accesses of a file in one time interval. Therefore, the value of r using the following formula can be calculated:

$$r = \left(N_f^{t+1} / N_f^t\right) - 1 \tag{4.1.1}$$

Assume t is the number of passed intervals, and N_f^t indicates the number of access for the file f at time interval t, then we get the sequence of access numbers:

$N_f^0\ N_f^1\ N_f^2\ N_f^3\ \ldots\ N_f^{t-1}\ N_f^t$

Therefore, there are $t-1$ time intervals, and each time interval has a growth or decay rate in number of accesses of a file. So according to the exponential growth/decay model, the equation can be written as in the following:

$r_0 = \left(N_f^1 / N_f^0\right) - 1,$

$r_1 = \left(N_f^2 / N_f^1\right) - 1,$

$r_2 = \left(N_f^3 / N_f^2\right) - 1,$

$r_{t-1} = \left(N_f^t / N_f^{t-1}\right) - 1$ (4.1.2)

Therefore the average rate for all intervals is:

$r = \sum_0^{t-1} r_i / t - 1$ (4.1.3)

Having known the average accessed rate (growth or decay) for a file during the past intervals, the number of access for the upcoming time interval can be estimated, which is termed as the File Lifetime, and this is computed as in the following equation:

$File\ Lifetime = N_f^t \times (1 + r)$ (4.1.4)

In order to avoid extreme cases where the growth or decay rate is equal to infinity, it is assumed that all files have been accessed for at least once. Using the data that is provided in Figure 4.2, an example to explain the concept of the strategy will now be presented. In this example there are four time intervals (t1, t2, t3, t4) with different number of accesses (NOA) of some data files.

First time interval (t_1)	
FileID	NOA
A	20
B	17
C	15
D	10

Second time interval (t_2)	
FileID	NOA
A	15
B	20
C	13
D	5
E	25

Third time interval (t_3)	
FileID	NOA
A	12
B	24
C	20
E	20

Fourth time interval (t_4)	
FileID	NOA
A	10
B	15
C	30
E	18

Figure 4. 1: An example of five files requests in four time

There are five different files (A, B, C, D, and E) accessed during the four time intervals (t_1, t_2, t_3, and t_4) In order to calculate the lifetime value of each file, the average growth/decay rate of each file during the four time intervals is calculated and then it is substituted into equations (4.1.3) and (4.1). Figure 4.3 shows the process of calculating the lifetime values for files A, B, and C. In the same way, the value of 1.76 was obtained as the number of accesses for file D, and 13.1 for file E. To this end the estimated number of accesses of a file is determined, which is known as File Lifetime (FL).

	t_1	t_2	t_3	t_4	t_5
A	20	15	12	10	N_A^5

- The average growth/decay rate of file A is:

$$r = \frac{(-0.25) + (-0.2) + (-0.16)}{3} = -0.21$$

- The estimated number of access of file A is:

$$N_A^5 = 10 * (-0.21 + 1) = 7.9$$

	t_1	t_2	t_3	t_4	t_5
B	17	20	24	15	N_B^5

- The average growth/decay rate of file B is:

$$r = \frac{0.18 + 0.20 + (-0.38)}{3} = 0.001$$

- The estimated number of access of file B is:

$$N_B^5 = 15 * (0.001 + 1) = 15.0$$

	t_1	t_2	t_3	t_4	t_5
C	15	13	20	30	N_C^5

- The average growth/decay rate of file C is:

$$r = \frac{(-0.13) + 0.54 + 0.50}{3} = -0.30$$

- The estimated number of access of file C is:

$$N_C^5 = 30 * (0.30 + 1) = 39.1$$

Figure 4. 2: An example of finding the file lifetime using Exponential model

Tracking Files' Behavior of Requesting a File (File Weight)

As mentioned in Chapter one, the data files that are used in this research are in the form of source code modality. Thus, there is a possibility to have some files that need other files in order to be executed or compiled. In other words, there might be a dependency relationship between files [152-154]. The dependency level differs from one file to another, i.e. the importance of a file to the environment is not the same.

The concern is to determine the importance of a file to the whole files system, which is termed as File Weight. The File Weight is computed by the following equation:

$$File\ Weight = \sum_{i=1}^{n} FL_i \times DL_i \quad (4.2)$$

where,

n: total number of files in a grid system,

FL: File Lifetime that has been calculated in the previous step, and

DL: dependency level of other files on the underlying file, and if there is no dependency then the DL is zero.

In order to understand how to calculate file weight, consider the following example. Suppose that the same files as in the example shown in Figure 4.2 are used, namely files A, B, C, D, and E, Figure 4.4 illustrates the dependencies between the five files, A, B, C, D, and E at time t-1. The present dependency relationships in Figure 4.4 would suggest that file B is more important than file A as there are three files (A, C, and E) that depend on file B while none exist for file A. Hence, the File Weight of files A, B, C, D, and E are obtained as follows:

$$\text{File Weight}(t, A) = 0$$
$$\text{File Weight}(t, \text{FileB}) = (8 \text{ x } 0.35) + (39 \text{ x } 0.33) + (8 \text{ x } 0.15) = 16.86$$
$$\text{File Weight}(t, C) = 0$$
$$\text{File Weight}(t, D) = 0$$
$$\text{File Weight}(t, \text{FileB}) = (8 \text{ x } 0.15) = 1.2$$

There is no focus on the dependency measurement, but rather it is on how to determine the file weight when a file is dependent on another.

.

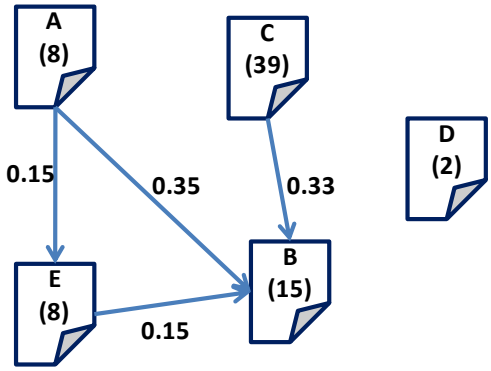


Figure 4. 3: Example of five dependence files

The proposed strategy evaluates data files in terms of users' perspective by combining information from user's behavior and file's behavior. With this, the File Value is computed as the following equation:

$$File\ Value(t,f) = FileLifetime(t,f) + FileWeight(t,f) \quad (4.3)$$

$FileLifetime$ (FL) and $FileWeight$ (FW) are used to compute the $File\ Value$ (FV) that is used as an indicator of the volume of demand on a file in a grid system, and the proposed replication algorithm will decide whether to replicate or delete a file.

Table 4.1 presents the result of computing the FV of the file in the previous example. Note that the most valuable file in users' perspective is file B.

Table 4. 1: Example of calculating file value for five files

File Name	**FileLifetime**	**FileWeight**	FV
A	8	0	8
B	15	16.86	31.86
C	39	0	39
D	2	0	2
E	8	1.2	14.97

4.2.3 Replica Creation and Deletion Strategy (RCDS)

In this strategy, DRCM triggers the replica creation or deletion function based on the file power of users' perspective, the file power of system's perspective, and a pre-defined threshold. The file power of users' perspective is represented by the $File\ Value$ that depends on FW and FL, and this is computed as follows:

$$FP_{users} = \frac{FV}{\sum_{\forall files} FV} \quad (4.4)$$

where,

FP: file power from user's perspective, and

FV: File Value.

On the other hand, the file power from the system's perspective is indicated by availability of a file in the system. This depends on the current number of replicas of the underlying file and this is computed as:

$$FP_{system} = \frac{NoC}{\sum_{\forall files} NoC} \quad (4.5)$$

where,

FP: file power from system's perspective, and

NoC: number of copies of the underlying file.

The main goal of this RCDM is to make a balance between users' perspective and system's perspective. In this context, making a balance occurs by increasing or decreasing the number of existing copies of files (FP_{system}) to meet the volume of demand on files within a grid (FP_{users}). In other words the balance can occur when:

$$FP_{users} = FP_{system} \quad (4.5)$$

However, a common scenario is when there are many requests but few replicas [155]. This is because storage capacities and other grid resources are limited. Therefore, in this thesis, the focus is on the most occurring scenario, so equation 4.5 will become as in the following:

$$FP_{users} = TH * FP_{system} \quad (4.6)$$

where, TH is the threshold value that determines how many percent the number of copies that are suppose to exist to meet the users request of the underlying file. The threshold value is specified by the system administrator as a percentage value, which varies according to the grid situation, such as the current bandwidth, the type of the running applications and jobs, and the workload of the system (number of jobs and number of files).

The DRCM calculates the FP in terms of users' and system's perspective by applying the equations 4.3 and 4.4 respectively, and computes the estimated number of replicas as follows:

$$\text{ENoR} = \frac{(\text{FP}_{\text{users}} - (\text{TH} \times \text{FP}_{\text{system}})) \times \sum_{\forall files} NoC}{\text{TH}} \quad (4.7)$$

where, ENoR: the estimated number of replicas.

There are three cases that may occur:

Case 1: if the $ENoR > 0$, then the system will replicate ENoR replicas of the underlying file,

Case2: if the $ENoR < 0$, then the system will delete ENoR of existing replicas, and

Case 3: if the $ENoR = 0$, then neither replication nor deletion is required.

In order to illustrate how the strategy works, consider the following example:

Assume a grid system has 15 files and their corresponding values and number of existing copies exists as shown in Table 4.2. Assume that the threshold value (TH) used is 50%, that means the FP_{users} should be two times the FP_{system} value.

Table 4. 2: Example of 15 files and their values and number of existing copies

File name	File value	Number of copies	File name	File value	Number of copies
File1	26	1	File9	25	1
File2	30	2	File10	22	4
File3	32	1	File11	13	1
File4	31	3	File12	9	2
File5	28	4	File13	11	3
File6	20	2	File14	8	1
File7	10	3	File15	17	1
File8	15	5	**Total**	**297**	**34**

The main concern here is to determine which file needs to be replicated and which file needs to be deleted. The first step is to calculate the power of each file in terms of users' perspective, and system perspective according to formulas (4.3), (4.4), and (4.7). For example, the power of File1 from users' perspective and system's perspective, and ENoR for File1 are computed as follows:

$$FP(File1)_{users} = \frac{26}{297} = 0.088$$

$$FP(File1)_{system} = \frac{1}{34} = 0.029$$

$$\mathrm{ENoR} = \frac{(0.088 - 2 \times 0.029) \times 34}{2} = 0.488 \cong 0.5 \cong 1$$

Due to the fact that number of replica values must be in the form of integer number, so the ENoR value is rounded up to the nearest integer. Therefore, the estimated number of replicas is 1, which means File1 needs to be replicated once. In the same way, all FP values and ENoR for each file are computed as shown in Table 4.3.

Table 4. 3: Example of calculating the ENoR of a file in DRCM

File name	Users power	System power	ENoR
File1	0.088	0.029	0.5
File2	0.101	0.059	-0.3
File3	0.108	0.029	0.8
File4	0.104	0.088	-1.2
File5	0.094	0.118	-2.4
File6	0.067	0.059	-0.9
File7	0.034	0.088	-2.4
File8	0.051	0.147	-4.1
File9	0.084	0.029	0.4
File10	0.074	0.118	-2.7
File11	0.044	0.029	-0.3
File12	0.030	0.059	-1.5
File13	0.037	0.088	-2.4
File14	0.027	0.029	-0.5
File15	0.057	0.029	0.0

The results from Table 4.2 show that File1 needs to be replicated by one copy as ENoR approximately equals to one, while three copies of File10 need to be deleted where its ENoR values approximately equal to three. Meanwhile, the ENoR for File2 and File9 approximately equal to 0, and therefore no action will occur as they are considered to be stable files. The rest of the files are in the same manner. To this end, there will be three lists of files, where the first list contains files that need to be replicated, the second list contains files that need to be deleted, and the third list contains files that require no further action.

4.2.4 Replica Placement Strategy (RPS)

The Replica Placement Strategy (RPS) finds the best location sites to place the newly created replicas. From the users' perspective, the best sites are located as close as possible to the sites that most potentially will request the underlying replicas to improve the geographical locality of the sites, while considering that the files that are requested by the sites are likely to be requested by nearby sites [144]. However, from the sites' perspective, the best sites are the ones that are located the farthest from the replication sites that never request the underlying replicas. Hence, choosing the best location sites depends on five parameters: Read cost, File Transfer Time, Site Workload, Replication Sites, and Dependence files distribution

Read Cost (RC)

RC is the cost of transferring a file from the underlying site to the remote sites [18], and this can be computed as:

$$RC = \frac{\sum_1^n FV_{s_i} \times FTT}{m} \quad (4.8)$$

where,

n: total number of sites in a grid,

m: number of sites that request a replica from the underlying site,

FV_{s_i}: file value with respect to the specific site s_i, which could be computed as:

$$FV_{s_i} = \frac{File\ Value}{NOR_{s_i}} \quad (4.9)$$

where,

NOR_{s_i}: number of requests for the popular file from a specific site s_i.

File Transfer Time (FTT)

FTT is the data transmission time, and depends on the size of the file and the current network bandwidth of the link between the two underlying sites. FTT is computed as in the following equation [46]:

$$FTT = \frac{File\ Size}{Bandwidt\ h} \quad (4.10)$$

Site Workload

The workload of a site is defined as the number of requests that can be satisfied by the underlying site [18, 151]. The candidate site should not exceed a specific amount of workload that has been assigned to it.

Replication Sites

A replication site is the site that is hosting the replica of the underlying file. Replication sites influences the candidate sites. The candidate site should be located as far as possible from the replication sites because of two main reasons: 1) the replication site itself never requests a replica that is already stored on it, 2) the load of user requests needs to be distributed.

Dependence Files Distribution

The placing of dependent files would affect the Read Cost (RC) of a popular file; accordingly, the decision making of placing the replicas will be affected. If the popular file depends on other files, therefore the RC of the popular file is the cost of transferring the popular file plus the cost of transferring other relevant files that are used by the popular files. Therefore, the distance between the relevant files and the candidate site should be as close as possible. In this case the RC of the file is computed using the following equation:

$$RC = RC(pf) + \sum_{d=1}^{m} FTT(file_d) * Dl(file_d) \quad (4.11)$$

where,

pf: the popular file, and

$file_d$: the file that the popular file depends on.

The proposed strategy, namely RPS, combines the five parameters together in order to make the decision on the placement of replicas, according to the following steps:

1. calculate the transfer time of each popular file by applying equation 4.10;

2. identify the sites that could be excluded from being candidates sites to hold the replicas, and those sites have the following characteristics:
   a. already stored the replicas in their storage elements (Replication Sites),
   b. already exceeded their maximum workload, and
   c. have a direct connection to replication sites;
3. determine whether the popular files have dependent files, if so then calculate the RC of each candidate site by applying equation 4.11, otherwise calculate the RC of each candidate site by applying equation 4.8;
4. for each popular file, select the sites from the list of candidate sites that provide minimum cost according to the number of replicas of each file; and if two files have the same site that provide minimum cost, and that site cannot accommodate both files, then prioritization of files is required, i.e. there is a need to decide which file will be accommodated in the underlying site. As mentioned previously, the main concern is to minimize the overall RC, so in order to do so, an examination of the next sites of the files that provide minimum RC (alternative sites) is performed. In other words, the author wants to check which file gets more RC when it moves to an alternative site, by calculating the Move Cost (MC) of the competitor files. The file that gets a lower MC value will be stored in its alternative site. The MC could be computed by the following equation:

$$\mathrm{MC}(\mathrm{file_i}) = \mathrm{RC}_{\mathrm{alternative\ \ site}}(\mathrm{file_i}) - \mathrm{RC}_{\mathrm{underlying\ \ site}}(\mathrm{file_i}) \quad (4.12)$$

In order to understand this strategy, consider the following example: suppose that there are eight sites with the bandwidth between sites as shown in Figure 4.4.

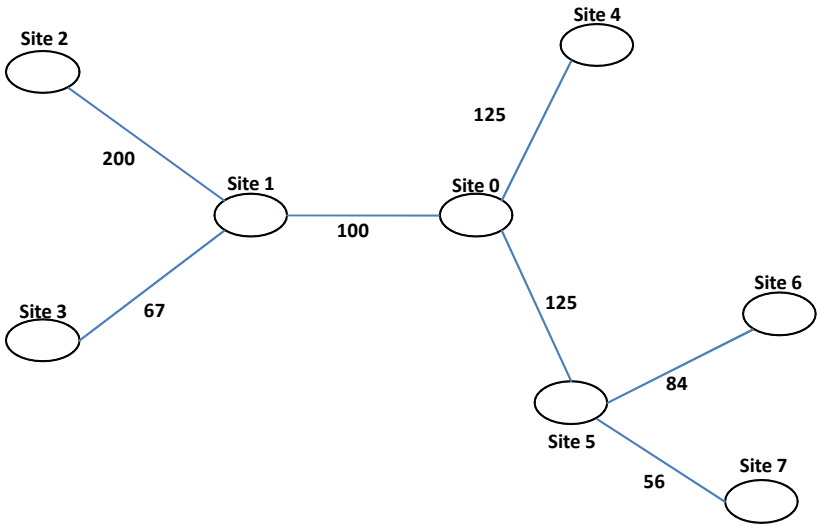


Figure 4. 4: A grid network consists of eight sites and network bandwidth

It is supposed that new replicas of the files File1, File2, and File3 with size = 1000MB, 400MB, and 700MB respectively need to be created by the system and hence requires site locations for an appropriate number of replicas of each file as follows: File1 *one replica*, File2 *two replicas*, and File3 *one replica*. After computing the FTT of each file by applying equation 4.10, the graph becomes as shown in Figure 4.5, Figure 4.6, and Figure 4.7. Number of accesses from each site to the file is shown within the parentheses. Suppose that *site5* already stores one replica of *File1*, *site6* stores one replica of *File2*, and *site1* stores one replica of *File3*.

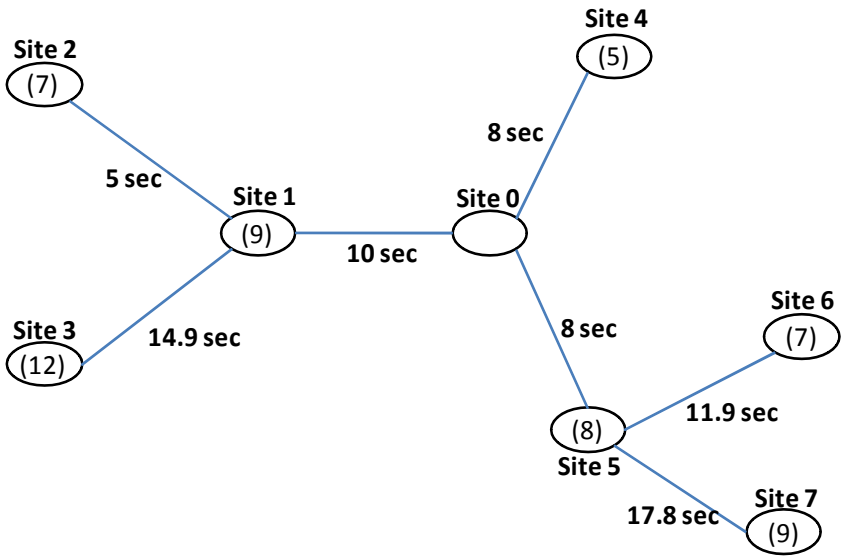


Figure 4. 5: A grid network consists of eight sites and their transfer time of File1

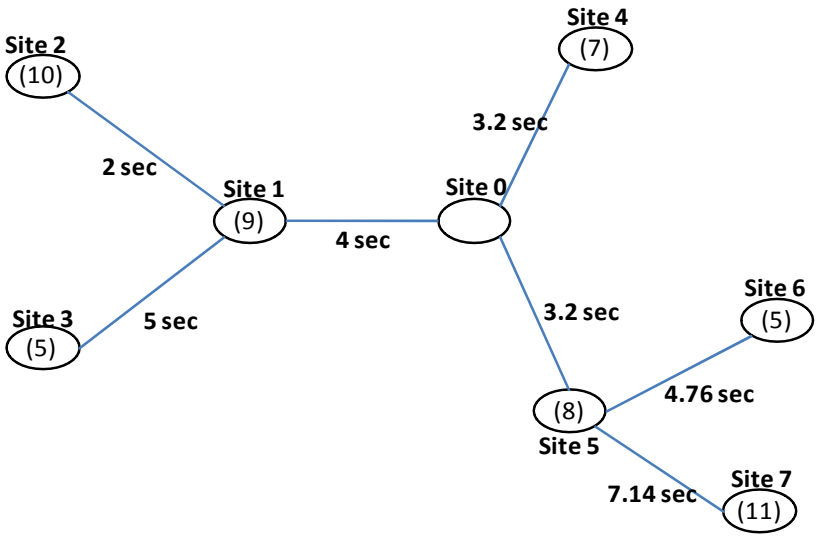


Figure 4. 6: A grid network consists of eight sites and their links that represent the transfer time of File2

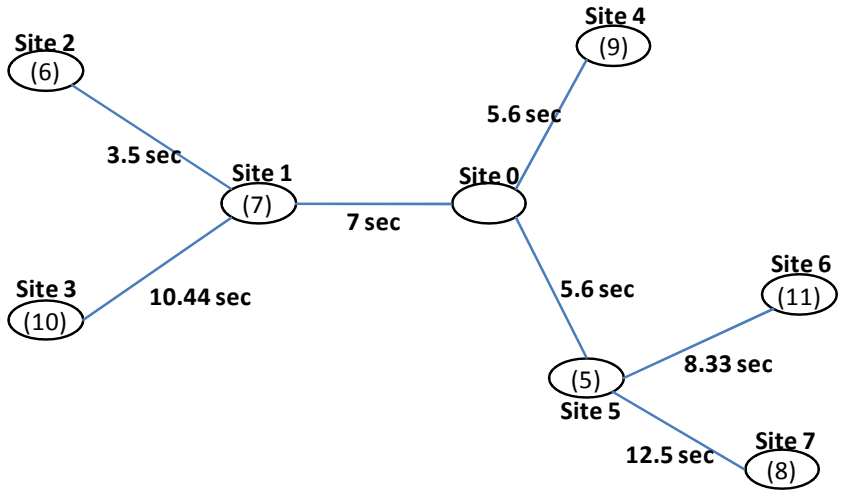


Figure 4. 7: A grid network consists of eight sites and their links that represent the transfer time of File3

Assume that *site4* will exceed its maximum workload if it hosts File1. Therefore, the following sites are excluded from being candidates to host File1:

Site4: because it already exceeds the maximum workload. Furthermore, *Site4* requests the replica from the nearest location, which is *Site5*.

Site5: because it has already stored the replica of *File1* on it.

Site6 and *Site7:* because they request the replica from the nearest location, namely *Site5*.

The RC for the candidate sites for *File1* are computed by applying equation 4.8, as shown below:

RC for site2(File1) = $\frac{(9\times5)+(12\times19.9)}{2} = 136.7$

RC for site1(File1) = $\frac{(9\times14.9)+(7\times19.9)}{2} = 141.4$

RC for site3(File1) = $\frac{(7\times5)+(12\times19.9)}{2} = 106.9$

Assume that *Site4* is able to host *File2* as it will not exceed its maximum workload. Therefore, the following sites are excluded from being candidates to host File2:

Site6: because it has already stored the replica of *File1* on it.

Site5 and *Site7:* because they request the replica from the nearest location, namely *Site6.*

The RC for the candidate sites for *File2* are computed by applying equation 4.8, as shown below:

RC for site2(File2) = $\frac{(9\times2)+(5\times7)+(7\times9.2)}{3} = 31.8$

RC for site1(File2) = $\frac{(10\times2)+(5\times5)+(7.2\times7)}{3} = 39.1$

RC for site3(File2) = $\frac{(9\times5)+(10\times7)+(12.2\times7)}{3} = 63.4$

RC for site4(File2) = $\frac{(10\times9.2)+(9\times12.2)+(9\times7.2)}{3} = 72.6$

The RC for the candidate sites for *File3* are computed by applying equation 4.8, as shown below:

RC for site5(File3) = $\frac{(9\times11.2)+(11\times8.33)+(8\times12.5)}{3} = 97.4$

RC for site6(File3) = $\frac{(5\times8.33)+(8\times20.83)}{2} = 104.1$

RC for site7(File3) = $\frac{(5\times12.5)+(11\times20.83)}{2} = 145.8$

Thus, the RC for each site can tabulated, as shown in Table 4.4. The intersection of the Sites in the rows and the Files in the column is the RC for the site of a certain file.

Table 4. 4: RC of three files for eight sights

	File1	File2	File3
$Site_0$	/	/	/
$Site_1$	136.7	31.8	/
$Site_2$	141.9	39.1	/
$Site_3$	106.9	63.4	/
$Site_4$	/	72.6	/
$Site_5$	/	/	97.4
$Site_6$	/	/	104.1
$Site_7$	/	/	145.8

According to the information in Table 4.4, the implemented strategy will decide to place *File1* in *Site3*, as it has the minimum RC, two copies of *File2* in *Site1* and *Site2*, and one copy of *File3* in *Site5*.

Nevertheless, if RC for *Site1* of *File1* is 95, *Site1* would provide the minimum RC for *File1* and *File2*. Thus, *File1* and *File2* have to be prioritized by applying equation 4.12 that computes the MC of the files. In this case, there is a need to identify the alternative site for *File1* and *File2*, which is *Site3*.

MC(File2) = RC for Site3(File2) – RC for Site1(File2) = 63.4 – 31.8 = 31.6

MC(File1) = RC for Site3(File1) – RC for Site1(File1) = 106.9 – 95 = 11.9

From the above result MC(File1) < MC(File2), *File2* will be placed in *Site1* and *File1* will be placed in *Site3*.

4.2.5 Determining the unwanted replicas placements

The unwanted replicas are the replicas of the file that get negative EnoR values and the system decides to delete them in order to reduce storage cost. Therefore, the

location sites from which the replica to be deleted are required. The calculations used here are the same as the ones used in RPS, but with some minor modifications. In locating the replicas to be deleted, the best site location is the site that has the maximum read cost instead of minimum read cost. In order to determine the best places for unwanted replicas, the sites that hold the replicas of that file are listed in descending order according to their read cost, and the sites with maximum read cost are chosen.

4.2.6 Replica Replacement Strategy (RRS)

RRS is applied when the selected site for placing the newly created replica has insufficient storage capacity for storing the underlying replica. RPS is a strategy that selects a victim file from the files that were stored in the target storage in order to free sufficient storage space for the underlying replica. The underlying replica is the file that has been chosen because it has high value compared to other files.

One approach by [44, 124] is to select the less valuable file to be the victim for deletion function, where the strategy may need to delete more than one file in order to store a new replica. For example, assume that there are nine files to be stored in one storage element with free storage space of 300 MB, as shown in Table 4.5, and assume that there is a replica with size 1000 MB that needs to be placed.

Table 4. 5: Example of nine files in one storage element and their corresponding FV and File Size

File name	File Value (FV)	File Size
File1	20	400
File2	18	500
File3	23	700
File4	25	1200
File5	24	1100
File6	27	1300
File7	19	900
File8	22	800
File9	28	1500

In the above example, the victim file is File2, as it has the least FV. However, we still need to delete one more file, since the replica is of size 1000 MB. So the next victim file is File7. With this, there is a need to delete two files in order to provide sufficient room for one file. In some cases, the number of victim files may reach up to five if they are of small sizes. This means that the system may lose five files that have been considered stable files in order to make room for one file.

To solve this problem, one suggestion is to delete the file that has the largest size. However, this approach is infeasible as in some cases such file may have the highest value among other files and the system would still require it.

Hence, in RRS, the two approaches are combined together, which means that two criteria to determine the victim file are considered, namely the file value (FV) and the storage cost (SC). To understand how the two approaches are combined, information in Table 4.5 is again referred to.

First of all there is a need to know how much storage capacity is needed to make room for the underlying replica, and this is done by applying the following equation:

$$RS = File\ Size - Free\ Space \quad (4.13)$$

where, *RS*: the required space to host the underlying replica

The files that have their sizes equal or more than the RS (Targeted Files) are selected. Therefore, the targeted files are sorted according to their values. In the above example:

RS = 1000 – 300 = 700, so the sizes of targeted files have to be equal to or more than 700 MB, as shown in Table 4.6.

Table 4. 6: the targeted files for deletion function

File Name	File Value (FV)	File Size
File7	19	900
File8	22	800
File3	23	700
File5	24	1100
File4	25	1200
File6	27	1300
File9	28	1500

Thus, the best file for the deletion function (the victim file) which has the minimum file value is File7, i.e. the victim file.

4.3 DRCM Implementation

As stated previously in Chapter three, the DRCM has been implemented in OptorSim simulator; the general specifications of OptorSim have been discussed in Section 3.3. In this section we discuss specifically the integration of DRCM and other existing algorithms into OptorSim, moreover we proved the validity of our proposed algorithm.

4.3.1 Integration of DRCM into OptorSim

OptorSim is capable of simulating many areas of the grid and these areas can be divided into packages, where each of which contains a collection of related classes. The package diagram shown in Figure 3.4 describes those within OptorSim and their relations. Starting at the lowest level, the *optorsim.time* package deals with how time is measured within the simulation, while *optorsim.infrastructure* simulates the underlying grid infrastructure including the network, grid sites, and basic components of the site: computing Element and Storage Element. The P2P network and messaging system along with the auctioning process is contained in the *optorsim.auctions* package. The functionality of the replica management components including Replica Location Service is implemented in the *optorsim.reptorsim* package, while the replica optimization strategies are in the *optorsim.optor* package. *Optorsim* is the highest level package, that simulates the resource broker and users, and controls the GUI.

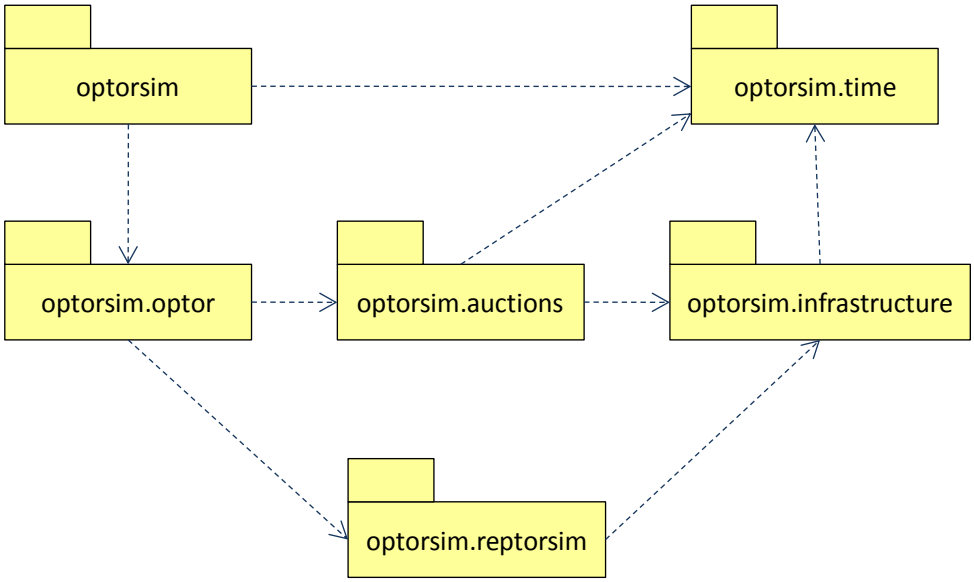


Figure 4. 8: UML package diagram of OptorSim

There are three replication algorithms that have already been implemented in OptorSim, namely LFU, LRU, and Economic algorithm. In this work, we implemented two algorithms namely LALW algorithm that considered as our benchmark, and DRCM which is the proposed algorithm. The DRCM and LALW along with other replication algorithms that have already been implemented in OptorSim, are written in Java and integrated into the *optorsim.optor* package of the simulator where it is termed as *DRCMOptimiser and LALWOptimiser*. The *DRCMOptimiser* and *LALWOptimiser* classes directly extend the skelOptor class that exists in *optorsim.optor* package as shown in Figure 4.9. The implementing classes and subclasses are shown as a UML class diagram in Figure 4.10. In general, the simulation works as follows: the process starts when users submit a job to the RB, which in turn searches for appropriate CE, and schedules the job to any CE by

following one of the scheduling algorithms defined in the parameter file. When the CE is ready to execute a job, it starts to process the files that are needed for the job.

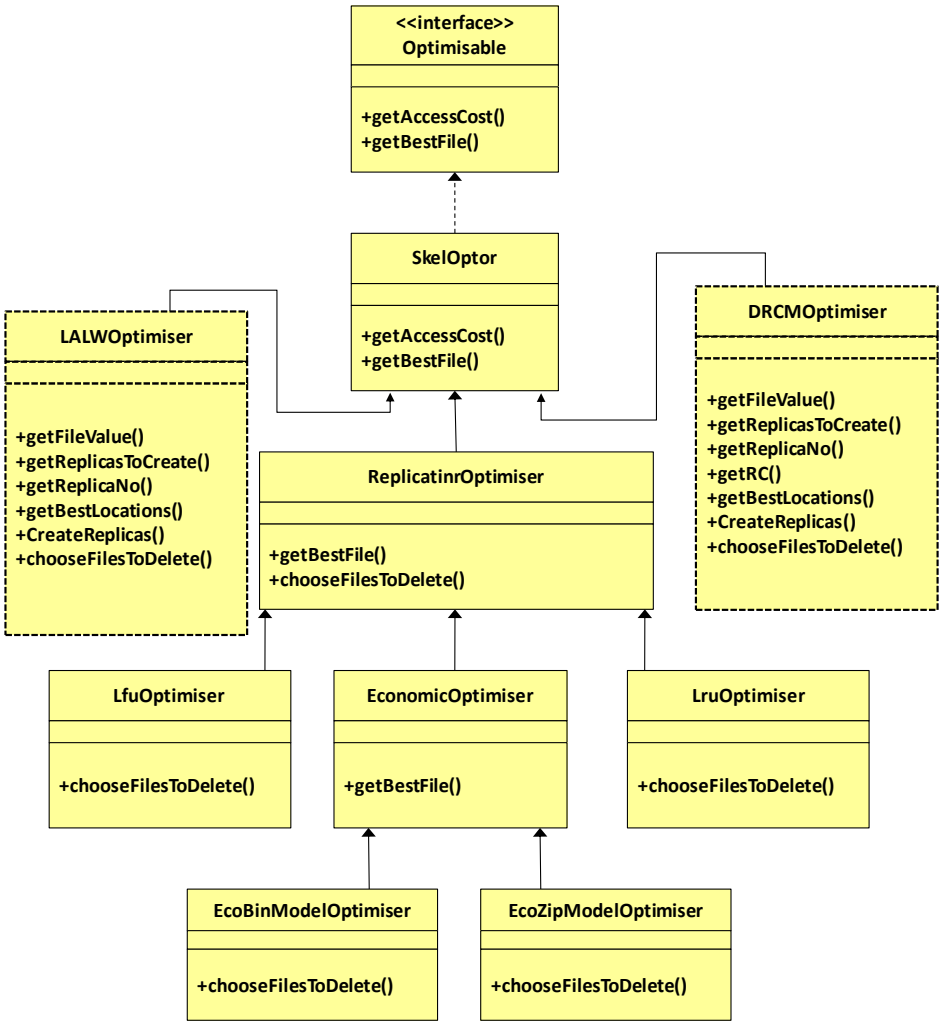


Figure 4. 9: UML class diagram showing the replication strategies classes in OptorSim

The order of processing the files is according to the access pattern defined in the parameter file. The CE then calls the local optimizer to find the best replica for the file. The CE then reads the file and processes it, before calling for the next file, and so on until all the files for the job have been processed. Based on OptorSim

architecture, each site has its own replica optimizer termed as local optimizer, and its main role is to find the best replica and replicate it in the local SE according to the chosen strategy. In this research, the Simple Optimizer is used as a local optimizer that finds the "best" replica of the required file but never replicates, as all files are read by remote I/O. The replication decision is made by the proposed algorithm, DRCM. In constant time interval, DRCM gets information of the files from Replica Catalogue (RC) (RC holds mappings of logical file names to physical file names [103]), evaluates the files in the system, and the accordingly makes the replication decision if it is necessary. When the replication process has been performed, the DRCM registers the new replica into the RC as shown in Figure 4.11.

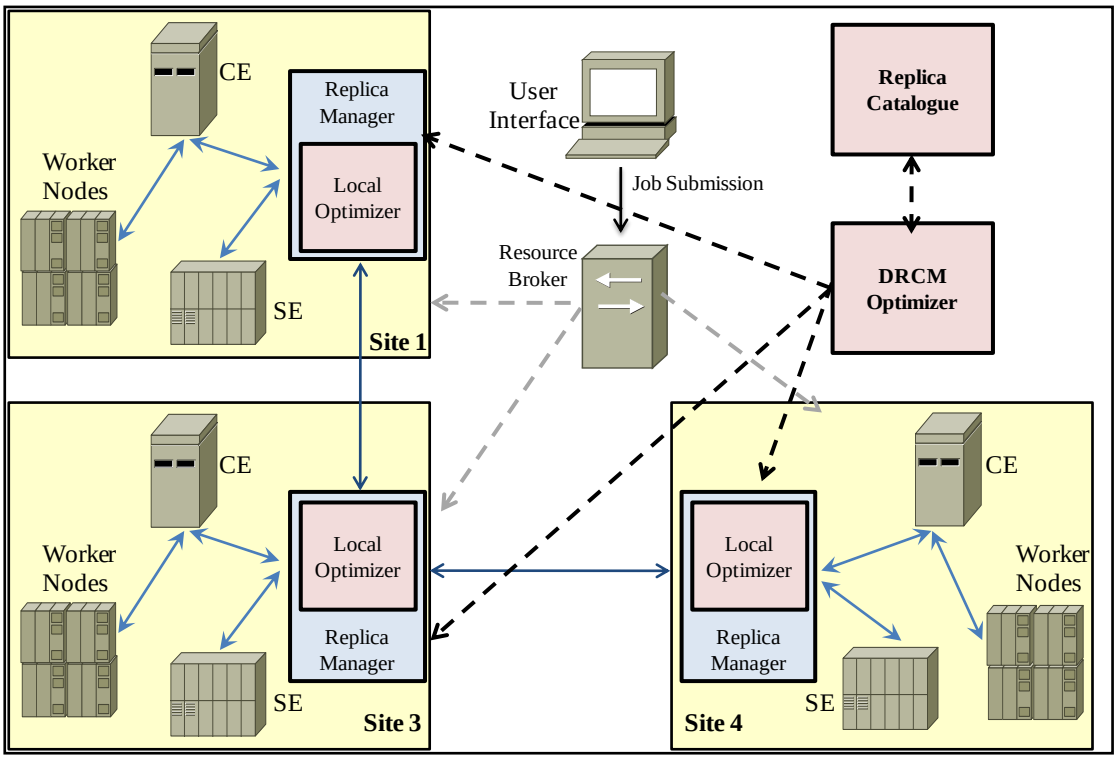


Figure 4. 10: Integration of DRCM into OptorSim

4.3.2 API of DRCM

In the previous section, we proposed a replica creation algorithm, in this section we present the main modifications that we made in order to implement and integrate the proposed algorithm.

The proposed algorithm is integrated into OptorSim by placing the algorithm in the Replica Optimizer. As other integrated algorithms, DRCM has its own storage element function that is used when the optimizer decides to make any file deletion or replication. There are two main classes need to be created, namely DRCM_Optimizer and DRCM_StoragrElement. DRCM_Optimizer direct the DRCM_StoragrElement to store replicas that are created by the optimizer as well as to remove files. the DRCM_StoragrElement will then execute the commands and thus stores or remove the particular files. Figure 4.13 and Figure 4.14 show the declarations of DRCM_Optimizer class and DRCM_StoragrElement class.

The DRCM_StoragrElement class includes the main methods of the DRCM that manipulate the stored data. Figure 4.15 shows the API of DRCM_StoragrElement class.

```
1   package org.edg.data.replication.optorsim.optor;
2
3   import java.util.*;
4   import java.util.Random;
5
6   import org.edg.data.replication.optorsim.infrastructure.*;
7   import org.edg.data.replication.optorsim.reptorsim.*;
8
9
10  public class DRCMOptimiser extends ReplicatingOptimiser {
11
12      protected DRCMOptimiser( GridSite site) {
13          super(site);
14      }
15
```

Figure 4. 11: Declaration of DRCM_StorageElement class

```
1  package org.edg.data.replication.optorsim.optor;
2
3  import java.util.*;
4  import java.text.DecimalFormat;
5
6  import org.edg.data.replication.optorsim.infrastructure.JobConfFileReader;
7  import org.edg.data.replication.optorsim.infrastructure.DataFile;
8  import org.edg.data.replication.optorsim.infrastructure.GridSite;
9  import org.edg.data.replication.optorsim.infrastructure.OptorSimParameters;
10 import org.edg.data.replication.optorsim.infrastructure.GridContainer;
11 import org.edg.data.replication.optorsim.infrastructure.*;
12 import org.edg.data.replication.optorsim.reptorsim.ReplicaManager;
13 import org.edg.data.replication.optorsim.reptorsim.*;
14
15
16 public class DRCMStorageElement extends AccessHistoryStorageElement{
17
18
19    public DRCMStorageElement(GridSite site, long capacity) {
20       super(site, capacity);
21    }
```

Figure 4. 12: Declaration of DRCM_Optimizer class

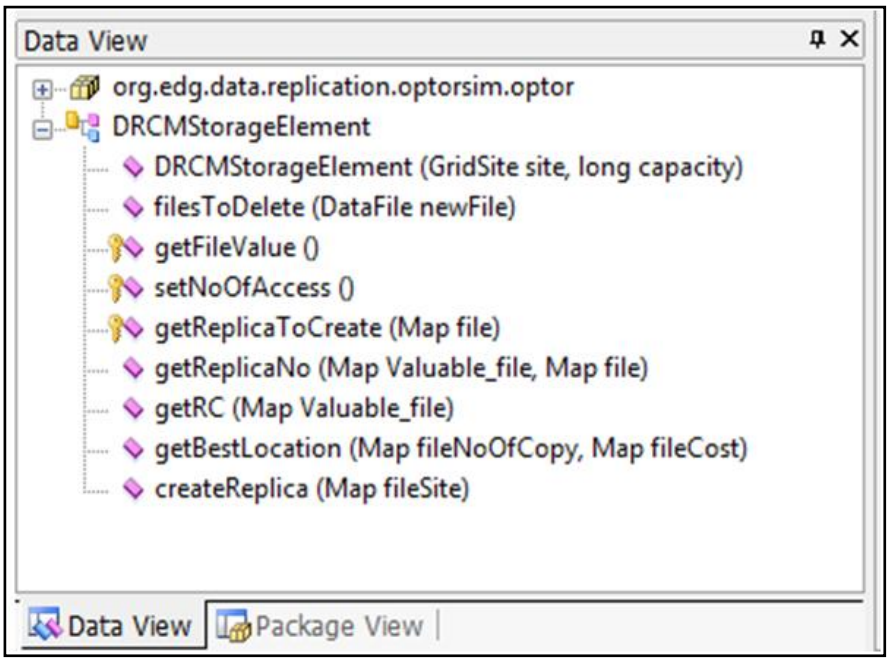


Figure 4. 13: The main methods of DRCM_StorageElement class

DRCM needs to be included in the parameters file so that it can be selected for the testing purpose. To do so, we modified the OptimiserFactory class and StorageElementFactory class. Figure 4.16 and Figure 4.17 show the modifications that have been done in both classes. The first five optimizers are already implemented in OptorSim, we added LALW_Optimizer and DRCM_Optimizer.

```
50  public StorageElement getStorageElement( GridSite site, long capacity) {
51
52      OptorSimParameters params = OptorSimParameters.getInstance();
53
54      switch( params.getOptimiser()) {
55
56          case SIMPLE_OPTIMISER:
57            return  new SimpleStorageElement( site, capacity);
58
59          case LRU_OPTIMISER:
60            return  new LruStorageElement( site, capacity);
61
62          case LFU_OPTIMISER:
63            return new  LfuStorageElement( site, capacity);
64
65           case ECO_MODEL_OPTIMISER:
66            return new  EconomicBinomialStorageElement( site, capacity);
67
68          case ECO_MODEL_OPTIMISER_ZIPF_BASED:
69            return  new EconomicZipfStorageElement( site, capacity);
70
71          case LALW_OPTIMISER:
72            return  new LALWStorageElement( site, capacity);
73
74          case DRCM_OPTIMISER:
75            return  new DRCMStorageElement( site, capacity);
76
77           default:
78            System.out.println("You have picked a non-existent optimiser, please try again.");
79            System.exit(1);
80          }
```

Figure 4. 14: The modifications of StorageElementFactory class

```
32  public static Optimisable getOptimisable( GridSite site) {
33
34      OptorSimParameters params = OptorSimParameters.getInstance();
35
36      switch( params.getOptimiser()) {
37
38      case SIMPLE_OPTIMISER:
39          return new SimpleOptimiser(site);
40
41      case LRU_OPTIMISER:
42          return new LruOptimiser(site);
43
44      case LFU_OPTIMISER:
45          return new LfuOptimiser(site);
46
47      case ECO_MODEL_OPTIMISER:
48          return new EcoBinModelOptimiser(site);
49
50      case ECO_MODEL_OPTIMISER_ZIPF_BASED:
51          return new EcoZipfModelOptimiser(site);
52
53      case LALW_OPTIMISER:
54          return new LALWOptimiser(site);
55
56      case DRCM_OPTIMISER:
57          return new DRCMOptimiser(site);
58
59      default:
60          System.out.println("You have picked a non-existent optimiser, please try again.");
61          System.exit(1);
62      }
```

Figure 4. 15: The modifications of OptimiserFactory class

4.3.3 Validation of DRCM

There are several methods that mentioned in [141] presented by Robert Surgent which is considered as a slandered reference for simulation modeling at an introductory level, a verity of techniques for validation are outlined, we used the relevant technique to the case of Replication algorithms. We used three methods to validate the DRCM, as follows:

Fixed Value Test: Fixed values of the simulation parameters are chosen (number of access, dependency level set to 0), and very simple topology such that expected results can be calculated manually, and then compared with actual results.

In our case we have used simple topology consists of two sites and we run 5 jobs, we have fixed the bandwidth and dependency level set to zero, and number of time intervals is one. We distribute only one copy of each file in the simulation. We invoked the DRCM after the 5 jobs have been submitted. We got number of access from the simulation and did manual calculations of which file suppose to be replicated and where. Then we used the existing function that is already implemented in Optorsim "listReplicas(String)", and we got the same files that expected to be .

Internal Validity: we run the simulation several times (around 20 times) to make sure that amount of variability is small. According to [149] the acceptable number of times for simulation is 10 times.

Comparison to Other Algorithms: We run the simulation and compare the results of DRCM to results of other (valid) algorithms.

4.4 Summary of Chapter

This chapter presented the design replication algorithm termed as Data Replica Creation Algorithm (DRCM). The main three objectives of the design are to minimize the job execution time, minimize the network bandwidth consumption, and minimize the storage element usage. In order to achieve the above objectives, DRCM encompasses three main parts:

i. *Replication Creation / Deletion Strategy* (RCDS)*:* in this part, the replicas that need to be created or deleted are determined, and number of created or deleted replicas is determined as well.
ii. *Replica Placement Strategy* (RPS)*:* in this part, the location sites from or to which the replica to be deleted or created are determined.
iii. *Replica Replacement Strategy* (RRS)*:* this stage is necessary when there is not enough space in the selected site to host the newly created replica, and then there is a need to determine the victim replica to be replaced with the newly created replica.

Numerical examples of the above strategies have been provided in this chapter to illustrate how each strategy in the proposed algorithm performs. We performed the modification of DRCM and implementation of the new algorithm (DRCM) in this chapter. However, we did not present the performance evaluation of this algorithm. The performance evaluation of this algorithm will attempt to show that the new algorithm performs better than the other existing algorithms. The next chapter will discuss the overall performance of the DRCM that is evaluated through simulated experiments.

CHAPTER FIVE
PERFORMANCE EVALUATION STUDY OF DRCM ALGORITHM

In order to evaluate the new DRCM, we conduct a comparative evaluation experiment between LALW algorithm that has been proposed in [156] and other existing algorithms (LFU, LRU, and Economy). This chapter presents a series of studies based on the EDG project and LALW topologies and configurations. With the aim of examining the efficacy of DRCM in different situations, a series of tests with their results are presented, with a verity of parameters that discussed in Section 3.4.2 (Number of jobs, types of schedulers, Access history length, and Storage sizes).

5.1 LALW-Based Topology

In this test scenario, DRCM is compared against LALW algorithm [37] and other existing algorithms (LFU, LRU, and Economy algorithm [107, 108]) using the configuration and parameters files as in LALW. The parameters used in this scenario are shown in Table 5.1, and the topology is shown in Figure 3.8 (Chapter three). This configuration has four clusters and each one has three sites. One site has the most capacity in order to hold all the master files at the beginning of the simulation. The others have a uniform size of 50GB. The network bandwidth is set as 100 Mbit/sec, while the connection bandwidth is 100 Mbps. We ran the simulation with 500 jobs.

A job is submitted to Resource Broker every 25 second. Resource Broker then submits to Computing Element according to an QAC scheduling algorithm.

There are 6 job types, and each job type requires specific files for execution. The order of files accessed in a job is sequential and is set in the job configuration file as an input to the simulation. The number of files in our simulation is 150, and a file size is 1GB.

Table 5. 1: LALW-Based topology: parameters settings

Parameter	Value
Number of Jobs	500
Scheduler	QAC scheduler
Site Policy	All Job Types
Access history length	1000000 ms
Storage metric (D)	0.67
Max. Queue Size	200
Job Delay	2500 ms
Dependency	No

The simulation results of DRCM and other algorithms are presented in Table 5.2.

Table 5. 2: LALW-Based topology: simulation results

Number of Jobs	Metrics	LRU	LFU	Economy	LALW	DRCM
500	MJET	4385	4154	4814	4013	3448
	ENU	47.13	47.41	36.72	33.11	31.95
	ASU	34.52	36.63	36.78	31.91	29.52
	CEU	18.87	20.23	24.52	25.91	26.19

In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed using the following standard equation:

$$Efficiency = \frac{metric\ \ value_{\ (existing\ \ \ system\ )} - metric\ \ value_{\ (DRCM\ )}}{metric\ \ value_{\ (existing\ \ \ system\ )}} \times 100 \qquad (5.1)$$

For example, DRCM outperforms LALW by 3.5% in ENU metric, LRU by 23.16% in MJET metric, and 14.48% in ASU metric. Table 5.3 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

Table 5. 3: LALW-Based topology: The efficiency results

Metrics	LRU	LFU	Economy	LALW
MJET	21.36%	16.99%	28.38%	14.08%
ENU	32.21%	32.61%	12.99%	3.50%
ASU	14.48%	19.41%	19.74%	7.49%
CEU	27.95%	22.76%	6.38%	1.07%

In what follows, we discuss and analyze in more details the results that presented in Table 5.3.

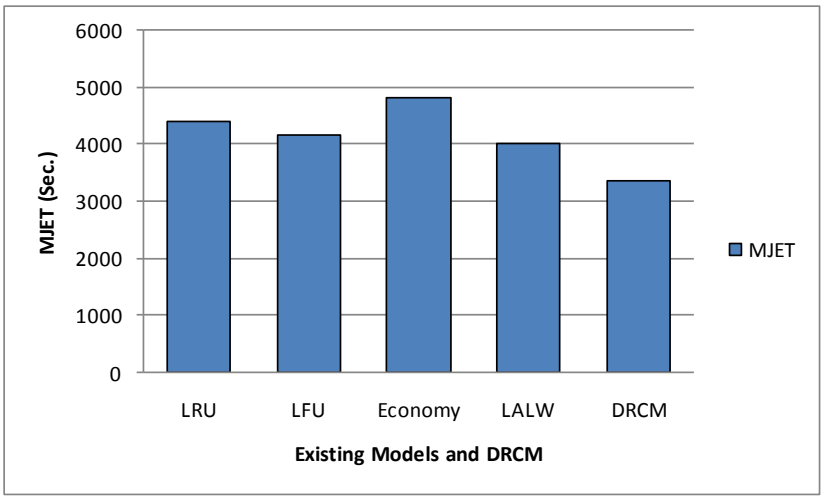


Figure 5. 1: The MJET of DRCM and existing Algorithms

A better algorithm is the algorithm that has less MJET. As shown in Figure 5.1, DRCM performs the best among the compared existing algorithms. DRCM consumes 16.05% less MJET compared to LALW, 30.02% over Economy, 18.89% over LFU, and 23.16% over LRU. This is due to the replication decision that has been made by DRCM, in which decides to replicate a group of valuable files at the same time (i.e. in one decision). As a result, replicas of popular files are spread in grid and increase the availability. On the other hand, LALW that replicates only one popular file at one decision. In addition, DRCM offers the decision includes the deletion of unwanted replicas, which helps in providing a free space in the storage element of sites. As a result, the need for invoking the replacement strategy will be decreased (save the time spent in determining the victim file). Thus, the replication process will be hastened, and spread the replicas as fast as possible.

Performing replication process affects the ENU metric, the ENU is calculated by the equation (3.3), the large value of $N_{replications}$ will increase the ENU value. The DRCM makes the decision of replication or deletion every constant time interval and start to replicate or delete the replicas. So at the beginning of the simulation, the number of replication is high, as a result, ENU increases. But, as time goes, the number of replication decreases, hence affects the value of ENU. Figure 5.2 shows the results of ENU metric, there was little difference between DRCM and LALW, although the DRCM gave slightly better network usage about 3% over LALW. LFU and LRU have the highest ENU, indicating that they are the worse algorithms, which make poor use of the network.

The effect of replica deletion process on ASU is quite pronounced, as shown in Figure 5.3 that will be discussed later in this section. Looking at other metrics, data in Figure 5.3 illustrates that DRCM uses the least amount of storage usage (ASU) by outperforming LALW by 7.49%. This is because in DRCM, the base of exponential decay varies based on the access rate of the file. Contrary to LALW approach which assumes that the base of exponential decay is constant and equals ½. That is, all files decay in the same rate regardless of its access rate. As a result, the declension rate of weight will be slower.

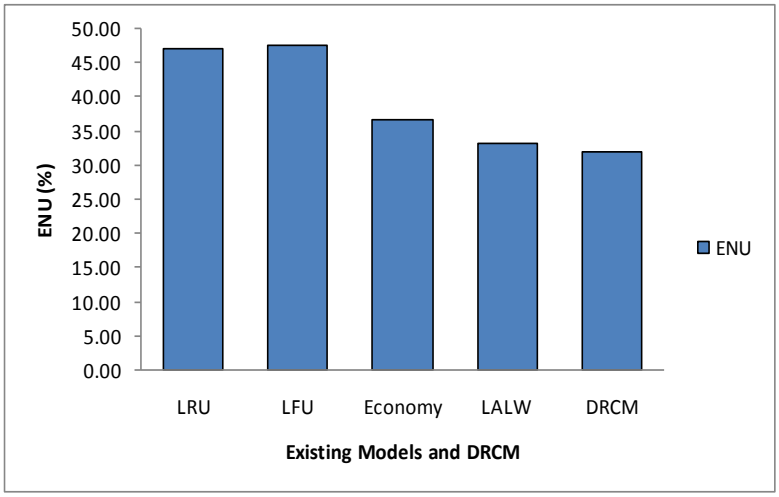


Figure 5. 2: The ENU of DRCM and existing Algorithms

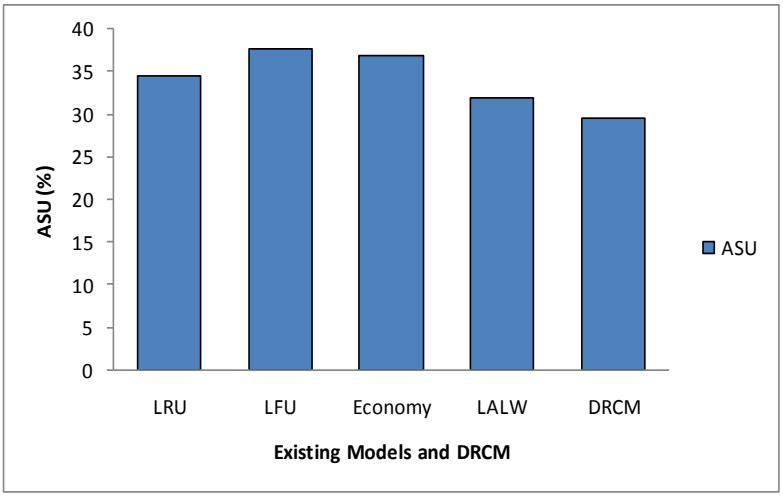


Figure 5. 3: The Storage Usage of DRCM and existing Algorithms

The impact of using scheduling algorithm could be clearly noticed in the results of CEU metric. In this scenario, Queue Access Cost (QAC) has been used as a job scheduler; it sends the job to the computing elements that are closed to the data. Figure 5.4 shows that DRCM performs the best in terms of CEU metric as it has the highest CEU value and outperform LALW by 1.07%. This is because DRCM distribute the replicas among the sites taking into account the workload of the sites in the grid and locations of existing replicas, which in turn drive the scheduling algorithm to make a balance while submitting the jobs, as they send the job to the computing elements that are close to the data.

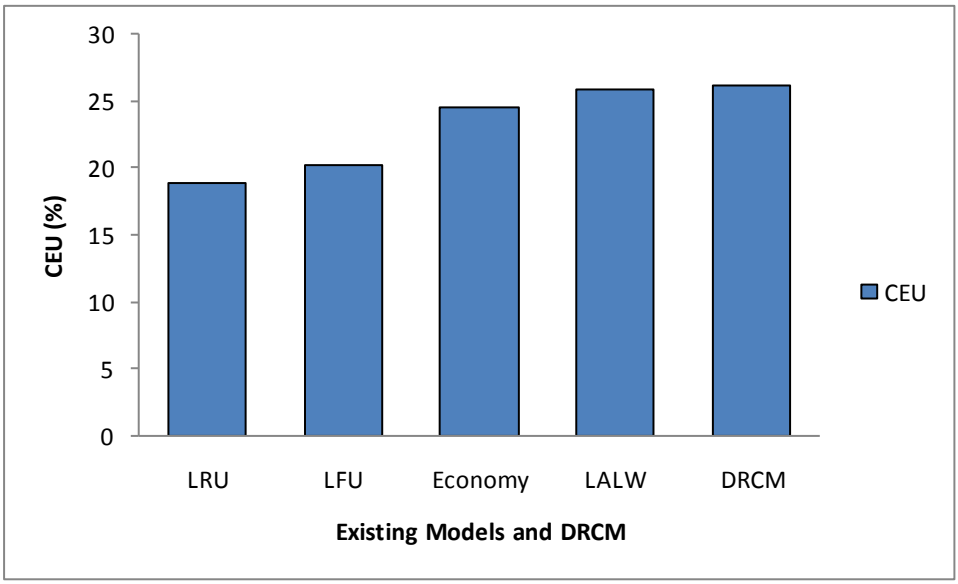


Figure 5. 4: The CE Usage of DRCM and existing Algorithms

5.2 LALW-Based Topology with File Dependency

In this scenario, we used the same topology, configuration and parameter files as in Section 5.2. However, this scenario includes the assumption that some of the files in

the grid depend on one another. Figure 5.5 illustrates an example of job configuration file that shows some of files dependency relationships. For example file *jpsi0* depends on three files, namely *jpsi4*, *jpsi8* and *jpsi9*. While file *jpsi6* depends only on one file which is *jpsi3*, and file *incelec4* is independent file (no dependency relationship with other files).

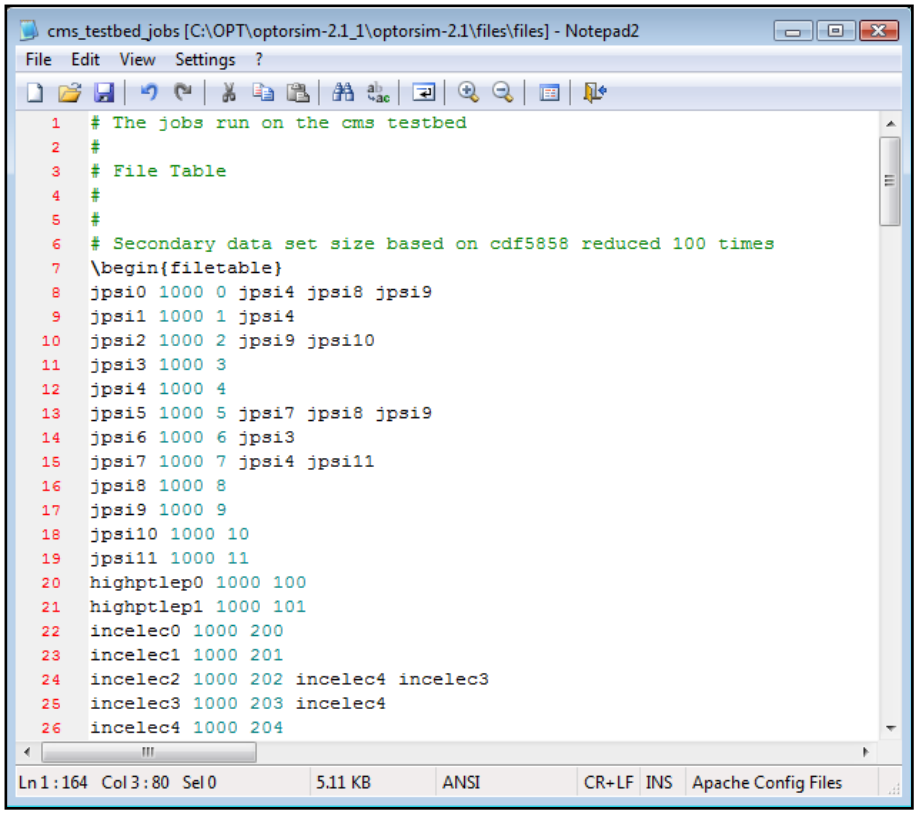


Figure 5. 5: An example of job configuration file showing dependency relationships of the files

We assume that the total dependency level of a file varies from 0% and 75%. That is, the minimum dependency level of a file is zero, in other words there is no dependency of the file, and the maximum dependency level of a file is 75%. For example, valid dependency values of file *jpsi0* should be something like 25% of file

jpsi4, 15% of file *jpsi8*, and 30% of file *jpsi9* so the total dependency level is 65% that is less than 75%.

The simulation results of DRCM and other algorithms are presented in Table 5.4.

Table 5. 4: LALW-Based topology with file dependency: simulation results

Number of Jobs	Metrics	LRU	LFU	Economy	LALW	DRCM
500	MJET	6777	6979	7992	7144	4792
	ENU	47.13	47.41	49.72	43.11	30.95
	ASU	34.22	35.33	35.28	30.01	29.12
	CEU	20.17	21.03	23.92	24.87	27.24

In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed by applying equation 5.1

Table 5.5 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

Table 5. 5: LALW test: Second test case: The efficiency results

Metrics	LRU	LFU	Economy	LALW
MJET	29.29%	31.33%	40.03%	32.91%
ENU	36.83%	36.09%	38.17%	30.23%
ASU	14.90%	17.58%	17.46%	2.97%
CEU	25.95%	22.80%	12.19%	8.70%

Table 5.4 provides results for the case with dependency relationships between files. Overall, there is a large increase in the mean job execution time when there is a dependency relationship between files. This can be attributed directly to the fact that

job execution time of a files equals to accumulated job execution times of all its dependence files, therefore, the time taken to complete a job will be increased much more. Of the five replication algorithms, as shown in Figure 5.5, the DRCM is the fastest strategy giving a projected saving in mean job execution time up to 32.91% over LALW, and 40.03%, 31.33%, and 29.29% over Economy, LFU, and LRU respectively.

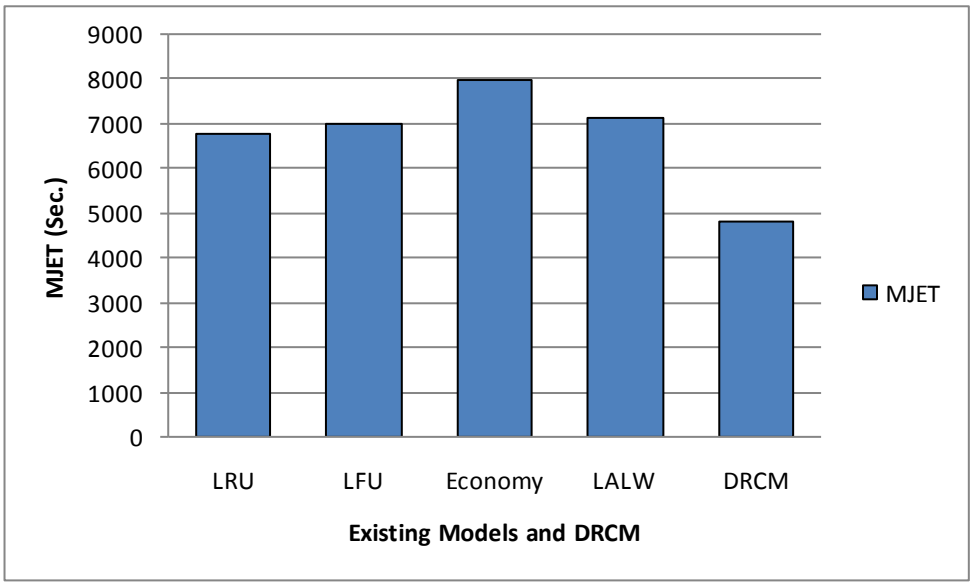


Figure 5. 6: The MJET of DRCM and existing Algorithms with dependency relationships

Figure 5.7 show that DRCM does make better use of the network. This is because DRCM replicates files based on their dependence level, as the important of the files derived from all of its dependence files. Thus, the Computing Element requires not only the original file that has been requested by the users, but also all of its dependence files. Congestion is therefore more likely, and if there is no consideration of the dependency issue, there is a repeatedly longer transfer time. The

results in Figure 5.7 show an advantage of using DRCM in terms of ENU. It has outperformed LALW by 30.96%, Economy by 38.17%, LRU and LFU by 36%. The DRCM distributes the newly created replicas to the sites such that they are as close as possible to their dependence files and as far as possible to the sites that already have the replica in its storage.

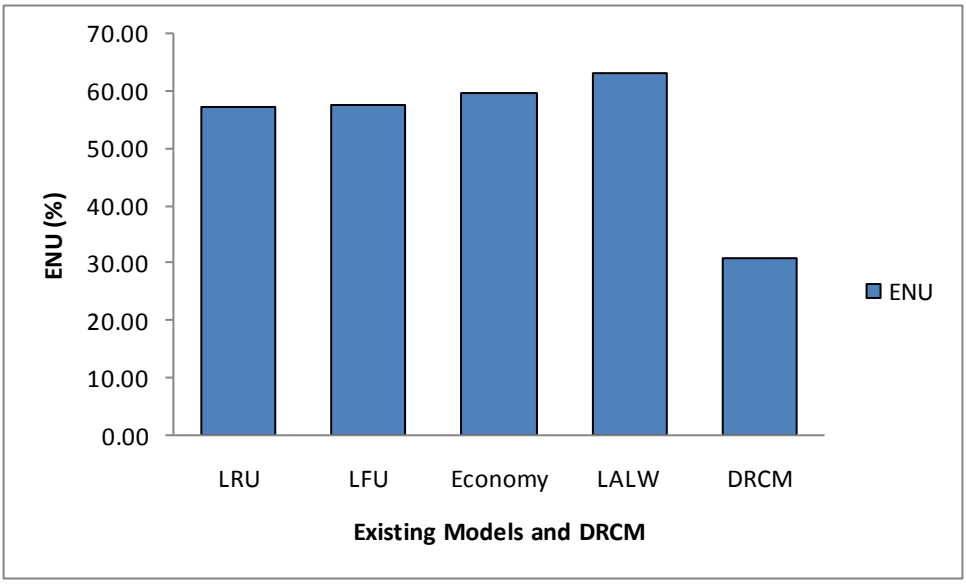


Figure 5. 7: The ENU of DRCM and existing Algorithms with dependency relationships

There is small difference in efficiency in terms of CE usage. As expected, the DRCM uses the highest CEU. This is for the same reason, as the job scheduler (QAC) submits the jobs accounting for data location and sends the jobs near to the required data. The advantage of using the DRCM over the existing algorithms is quite pronounced in Figure 5.8, DRCM outperform LALW by 8.70%, Economy by 12.19%, and LFU, LRU by 22.80%, 25.95% respectively.

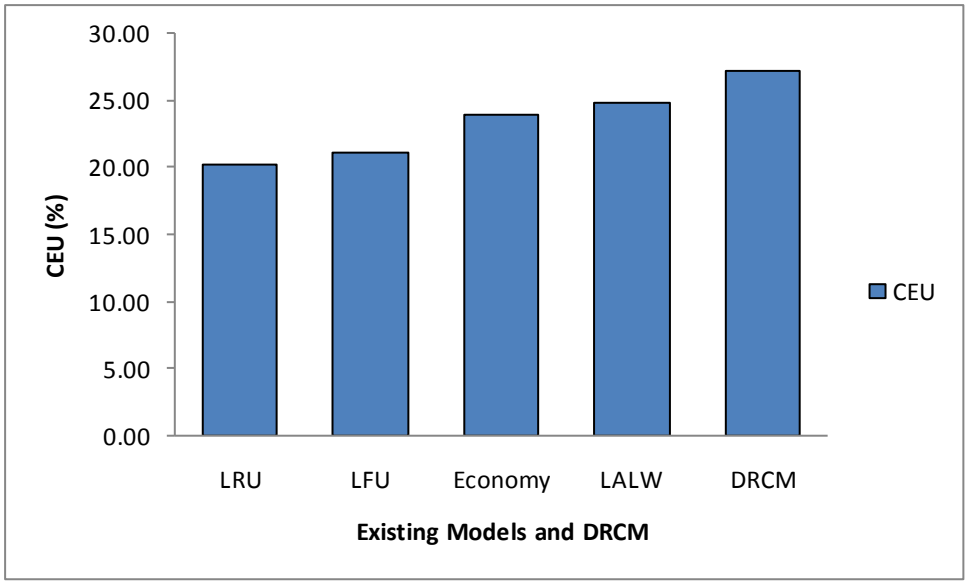


Figure 5. 8: The CEU of DRCM and existing Algorithms with dependency relationships

5.3 Analysis on Number of Access

As we discussed in Chapter two, LALW evaluates the files based on half-life algorithm that assumes the value of the files decay to half of its previous value. In order to evaluate the files in the current time interval, the file values of all of the previous time intervals are accumulated and termed as an Access Frequency of the file. As a result, the values of the file will be quite large even though they are not being accessed anymore. On the other hand, DRCM evaluates the files based on exponential growth/decay algorithm. In order to evaluate the files in current time interval, the average growth/decay rate is calculated and substituted in the exponential formula; refer to Figure 4.3 to see an example. Figure 5.9 shows comparative results between DRCM and LALW in evaluating 30 different files.

Compared with the values of recent number of access for the files, LALW generated large values. On the other hand DRCM generates values that are somehow close to the actual NOA

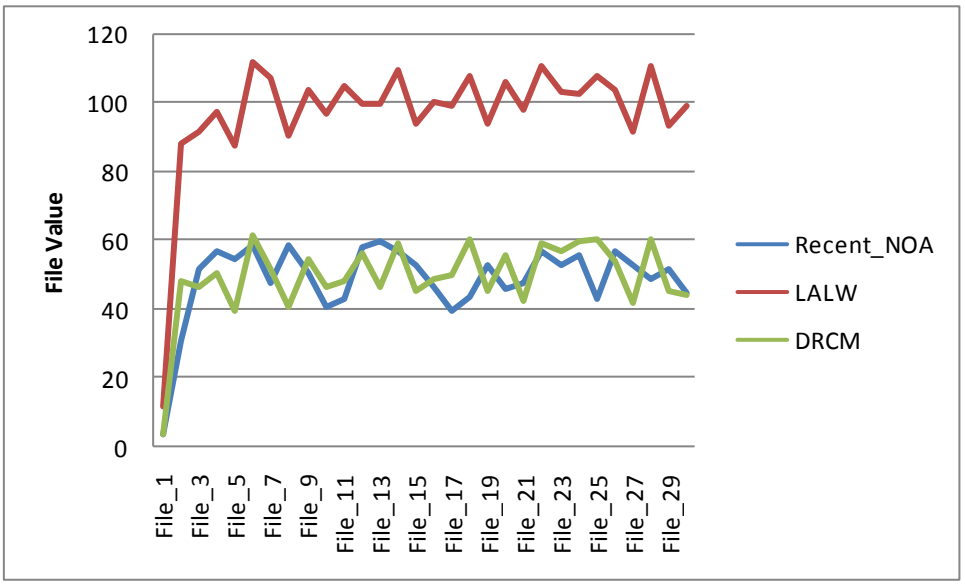


Figure 5. 9: NoA for 30 files: DRCM vs. LALW

5.4 Analysis on storage element usage

As a result of inaccurately evaluating of the files that been done by LALW, the declensional rate of the files will be slow, and the storage element will be compelled to store unnecessary files for long time due to their high values. Figure 5.10 illustrates an example of the decay rate of one file during 20 time intervals using DRCM and LALW algorithms. It is clearly seen that the declensional rate of the files using DRCM algorithm is much faster than LALW algorithm, therefore the unnecessary files that are not being accessed by the users will be quickly deleted to make a space for the newly created files.

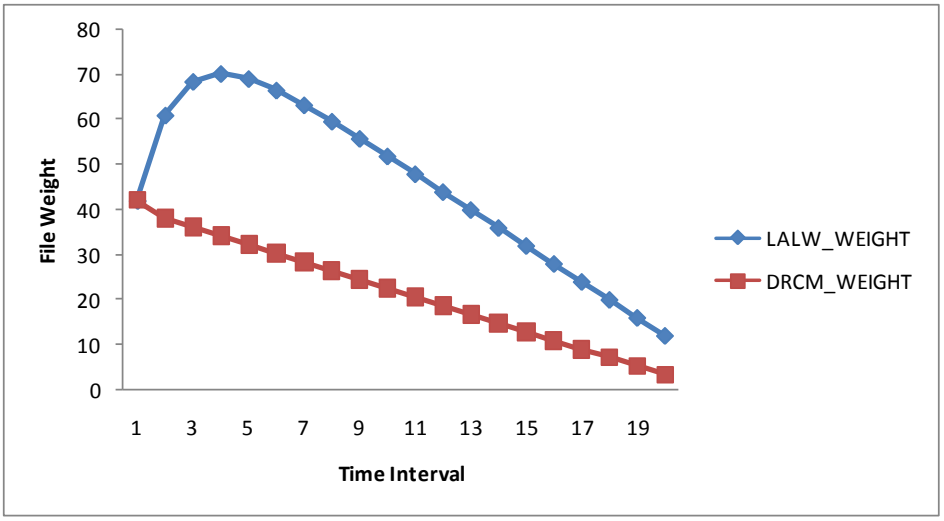


Figure 5. 10: Decay rate of one file during 20 time intervals: DRCM vs. LALW

5.5 Examining of the DRCM in Detail

While in Section 5.2, some results of running the simulation of a topology with parameters and configuration used in LALW test, this section presents a series of studies based on the EDG project resources and configurations. With the aim of examining the efficacy of DRCM in different situations, a series of tests with their results are presented, with a verity of parameters that discussed in Section 3.4.2 (Number of jobs, types of schedulers, Access history length, and Storage sizes).

5.5.1 Analysis: Number of Jobs

It is important to understand how the replication algorithms perform with increasing numbers of jobs on the grid [37]. Using the Queue Access Cost scheduler, with the other parameters as described in Section 3.4.2, the number of jobs submitted was varied from 200 to 4000 and the performance of the DRCM and other replication algorithms are measured. The basic settings and parameters used in this experiment

are shown in Table 5.6, and summary of results of workload test is shown in Table 5.7.

Table 5. 6: Workload test: parameters settings

Parameter	Value
Number of Jobs	200, 500, 1000, 2000, 4000
Scheduler	QAC scheduler
Site Policy	All Job Types
Access history length	1000000 ms
Storage metric (D)	0.67
Max. Queue Size	200
Job Delay	2500 ms

Table 5. 7: Workload test: simulation results

Number of Jobs	Metrics	LRU	LFU	Economy	LALW	DRCM
200	MJET	4582	4378	5543	4231	3792
	ENU	55.82	54.03	37.93	35.87	32.66
	ASU	34.58	33.96	36.73	33.13	28.91
	CEU	21.83	19.53	23.41	23.15	22.54
500	MJET	10011	9484	10992	9163	7692
	ENU	46.49	47.56	37.26	33.77	30.17
	ASU	36.70	37.41	37.97	35.71	29.52
	CEU	18.87	20.31	24.15	25.91	26.38
1000	MJET	16108	16180	22110	15351	13532
	ENU	43.42	43.21	37.19	32.88	27.54
	ASU	39.49	39.64	39.97	37.82	31.46
	CEU	25.34	25.60	28.27	30.25	32.62
2000	MJET	55567	56958	64026	54883	51189
	ENU	45.76	46.42	36.16	31.45	26.19

4000	ASU	40.63	40.64	40.64	39.63	33.11
	CEU	21.50	20.43	23.83	25.74	31.75
	MJET	108652	106979	134396	105629	103371
	ENU	47.83	47.53	34.73	30.19	24.37
	ASU	40.62	40.64	40.64	38.11	34.63
	CEU	23.96	23.88	26.27	28.11	31.91

From the simulation results shown in Table 5.7, the average of the values for each metric is computed to show the overall performance. The average results are shown in Table 5.8

Table 5. 8: Workload test: Average of simulation results

Metrics	LRU	LFU	Economy	LALW	DRCM
MJET	38984	38796	47413	37852	35915
ENU	47.87	47.75	36.65	32.83	28.186
ASU	38.40	38.46	39.19	36.88	31.53
CEU	22.30	21.95	25.19	26.63	29.04

In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed by applying equation 5.1. For example, DRCM outperforms LALW by 14.15% in ENU metric, LRU by 7.87% in MJET metric, and 18.02% in ASU metric. Table 5.10 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

Table 5. 9: Workload test: Efficiency results

Metrics	LRU	LFU	Economy	LALW
MJET	7.87%	7.43%	24.25%	5.12%
ENU	41.12%	40.97%	23.09%	14.15%

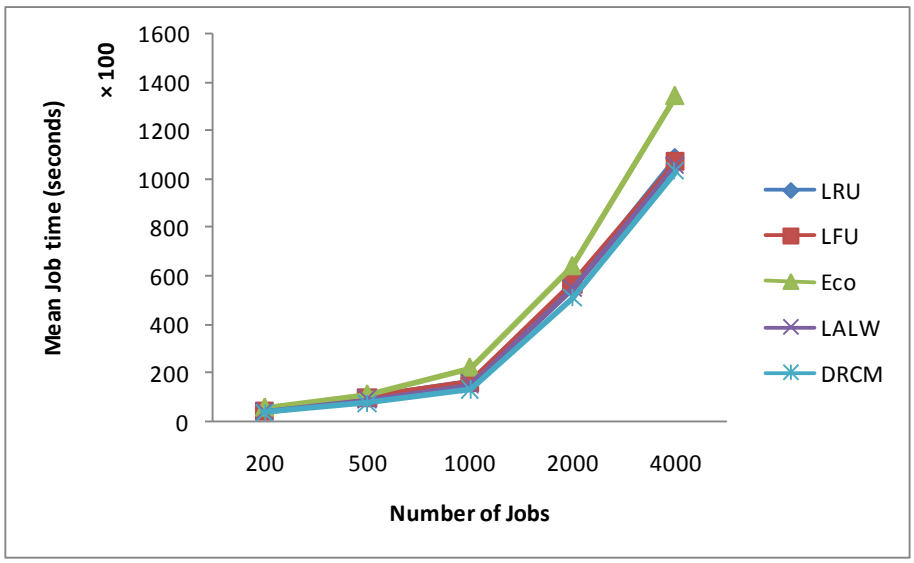
ASU	17.89%	18.02%	19.55%	14.51%
CEU	30.22%	32.30%	15.30%	9.04%

In what follows, we discuss and analyze the results that presented in Table 5.7.

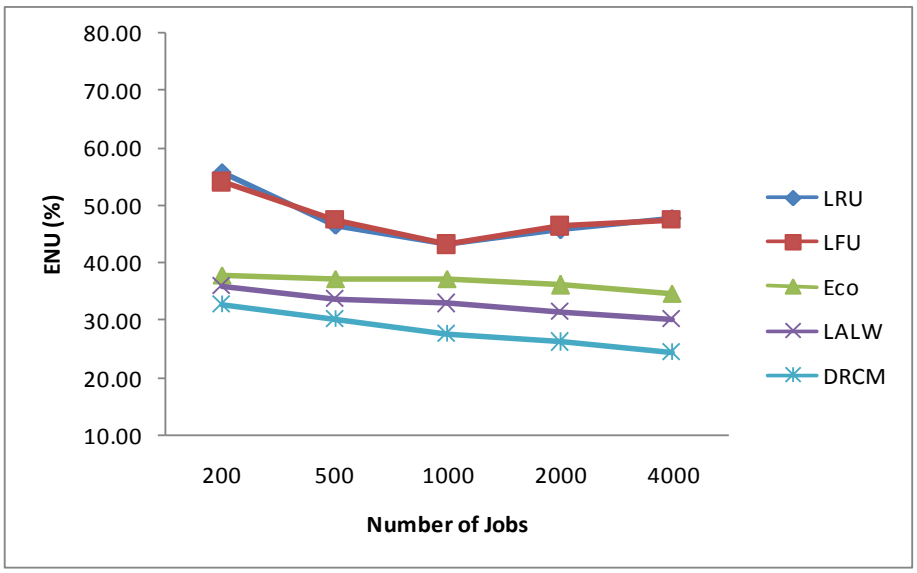
The results show a linear increase in the MJET as the number of jobs on the grid increases. This is because, as more jobs are submitted, the queues at the sites increase. If the job submission rate is higher than the grid's job processing rate, this build-up of queues is inevitable, and it is likely that this would also occur in a real grid. A better algorithm is a algorithm that has less MJET, as shown in Figure 5.11; DRCM performs the best among the other existing algorithms in the MJET metric. The DRCM is the fastest and improve the performance with MJET by 5.12% over LALW, 24.25% over Economy, and about 7 % over LRU and LFU.

Looking at the other metrics, LFU, LRU, and Economy algorithms give the highest storage usage (ASU) as they always replicate the files to the local storage element, followed by LALW. However, it is immediately obvious that using DRCM greatly improve the performance with the ASU, as it outperforms LRU, LFU, Economy, and LALW by 17.89%, 18.02%, 19.55%, 14.51% respectively. The results of ENU metric show a slight linear decrease as the number of jobs on the grid increases. This is because at the start of the simulation the queues are small, but they build up quickly while the files are copied around the grid. Once the replication process has established, the jobs can run faster and hence the queues decrease. The effective network usage meanwhile decreases gradually with increasing numbers of jobs because the amount of replication decreases over time. LRU and LFU have the

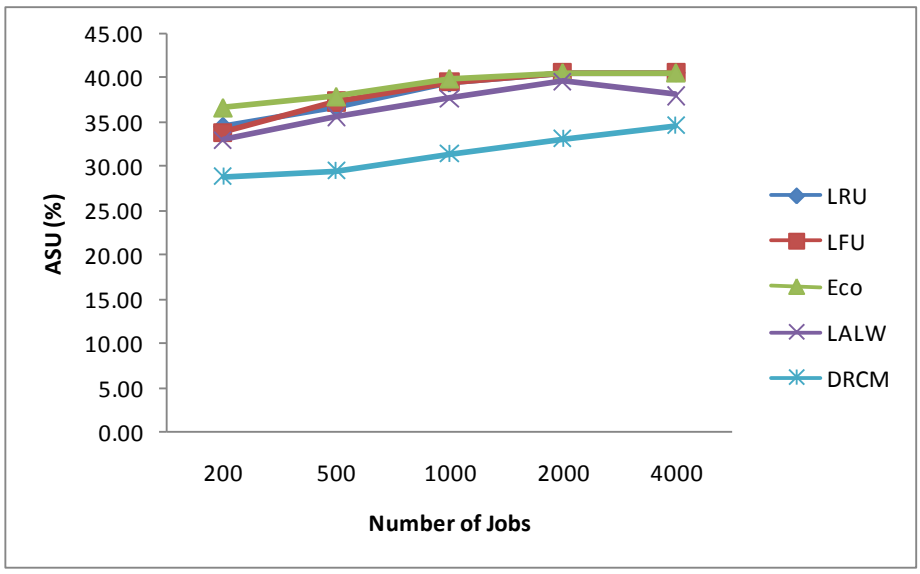
highest effective network usage, showing that they are poor at making replication decisions. The DRCM uses the lowest amount of network resources for all numbers of jobs because it makes better decision on which file should be replicated and which replicas each site should store. Looking at CEU metric, can be seen that the CE usage generally rises as the number of jobs increases, reflecting the heavier workload. However, there is an obvious drop between 1000 and 2000, this is because with the higher number of jobs, the scheduling algorithm is sending most of the extra jobs to a few sites from where the data are easily accessible, leading to more uneven distribution of jobs around the grid. The same trend, although less marked, there is a slight drop can also be seen with DRCM. This indicates that DRCM leads to make a good balance in the grid, this is because DRCM distribute the replicas among the sites taking into account the workload of the sites in the grid and places of the existing replicas, which in turn drive the scheduling algorithm to make a balance while submitting the jobs, as they send the job to the computing elements that are close to the data



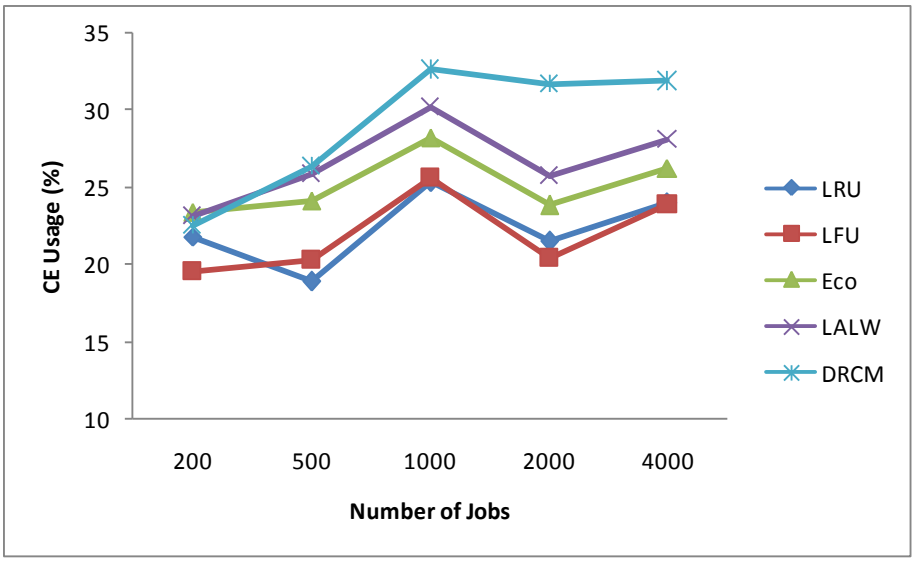
(a)


80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
ENU (%)
200
500
1000
2000
4000
Number of Jobs
LRU
LFU
Eco
LALW
DRCM

(b)


45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
ASU (%)
200
500
1000
2000
4000
Number of Jobs
LRU
LFU
Eco
LALW
DRCM

(c)



(d)

Figure 5. 11: (a) Mean job time, (b) ENU, (c) ASU and (d) CE Usage for different replication algorithms, with varying number of jobs.

5.5.2 Analysis: Types of Scheduler

The objective of this test is to investigate the performance of DRCM when different scheduler algorithms, namely random and queue access algorithm are used. The simulation was run with each of the scheduling algorithm, measuring the mean job time, ENU, and storage element usage for DRCM and other replication algorithms. The basic settings and parameters used in this experiment are shown in Table 5.10, and the summary of the results of this test is shown in Table 5.11.

Table 5. 10: Types of scheduler test: parameters settings

Parameter	Value
Number of Jobs	500, 1000

Scheduler	Random, QAC scheduler
Site Policy	All Job Types
Access history length	1000000 ms
Storage metric (D)	0.67
Max. Queue Size	200
Job Delay	2500 ms

Table 5. 11: Types of scheduler test: Simulation results

Number of Jobs	Job Scheduler	Metrics	LRU	LFU	Economy	LALW
500	QAC	MJET	10011	9484	10992	9163
		ENU	46.49	47.56	37.26	33.77
		ASU	36.70	37.41	37.97	35.71
		CEU	18.87	20.31	24.15	25.91
500	Random	MJET	32602	30407	30533	27185
		ENU	57.76	57.42	54.97	50.25
		ASU	40.64	40.64	40.64	36.67
		CEU	17.35	17.05	18.60	19.82
1000	QAC	MJET	16108	16180	22110	15351
		ENU	43.42	43.21	37.19	32.88
		ASU	39.49	39.64	39.97	37.82
		CEU	25.34	25.60	28.27	30.25
1000		MJET	60050	59056	57380	54420
		ENU	54.76	54.85	53.09	50.62
		ASU	40.64	40.64	40.64	38.16
		CEU	17.61	16.70	18.84	20.38

From the simulation results shown in Table 5.11, the average of the values for each metric is computed to show the overall performance. The averages are shown in Table 5.12.

Table 5. 12: Types of scheduler test: Average of simulation results

Metrics	LRU	LFU	Economy	LALW	DRCM
MJET	29692	28781	30253	26529	24084
ENU	50.60	50.76	45.62	41.88	35.97
ASU	39.36	39.57	39.80	37.09	30.91
CEU	19.79	19.92	22.47	24.09	25.11

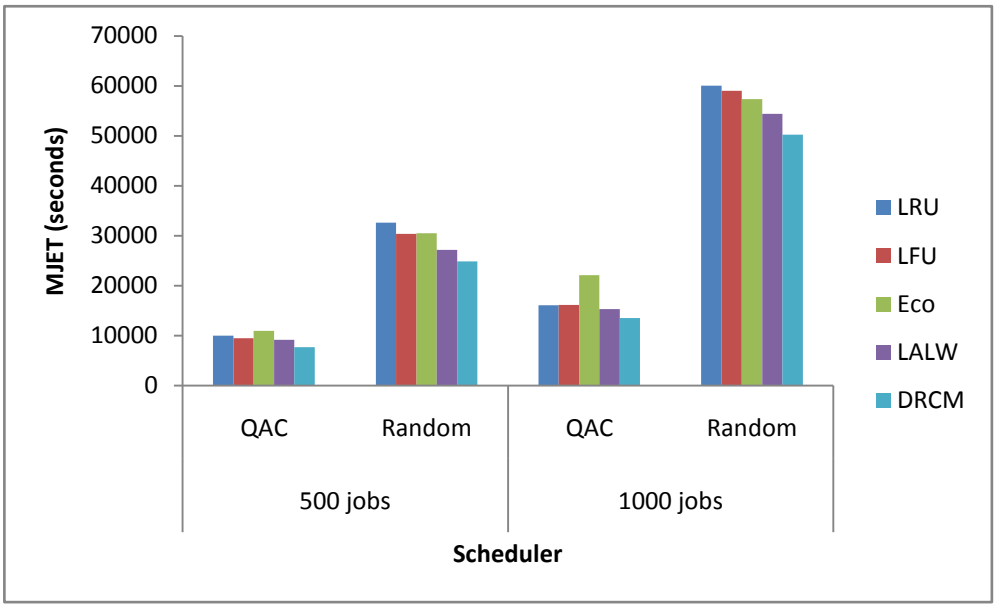
In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed by applying equation 5.1

Table 5.13 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

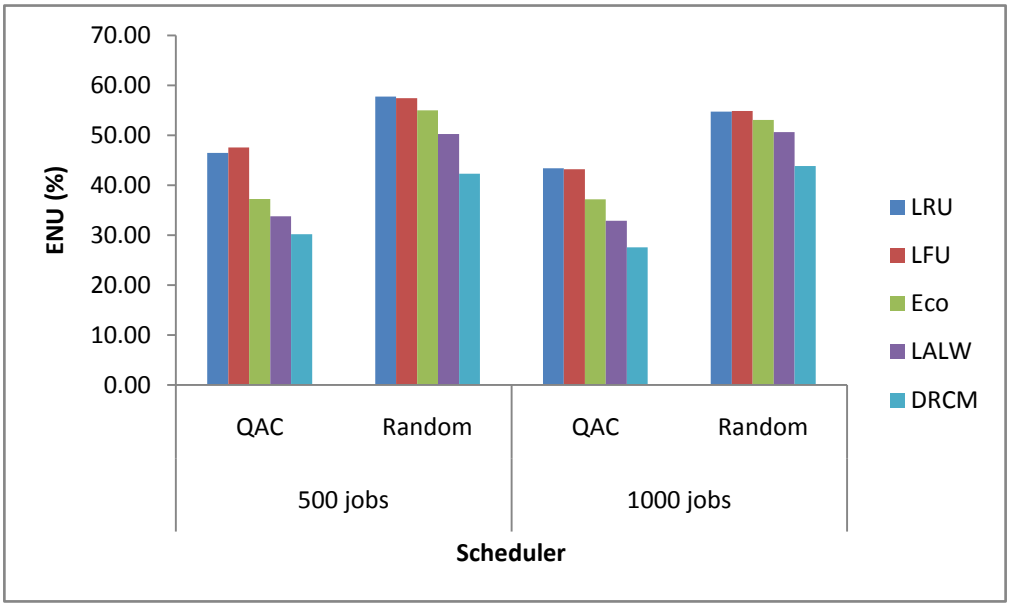
Table 5. 13: Types of scheduler test: Efficiency results

Metrics	LRU	LFU	Economy	LALW
MJET	18.89%	16.32%	20.39%	9.22%
ENU	28.92%	29.13%	21.16%	14.11%
ASU	21.46%	21.89%	22.33%	16.65%
CEU	26.88%	26.10%	11.78%	4.24%

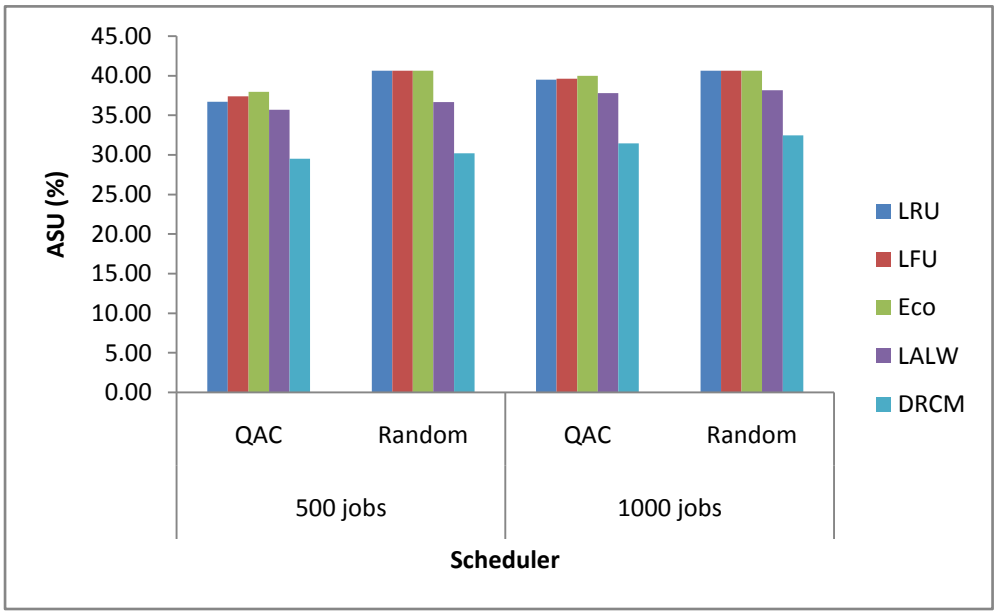
In what follows, we discuss and analyze the results that presented in Table 5.11.


70000
60000
50000
40000
30000
20000
10000
0
MJET (seconds)
QAC
Random
QAC
Random
500 jobs
1000 jobs
Scheduler
LRU
LFU
Eco
LALW
DRCM

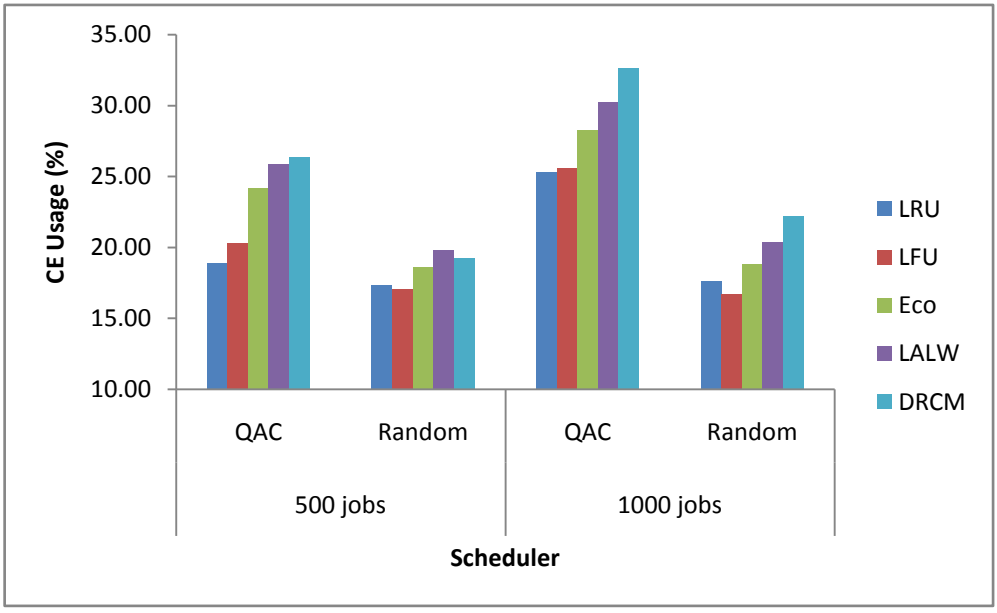
(a)


70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
ENU (%)
QAC
Random
QAC
Random
500 jobs
1000 jobs
Scheduler
LRU
LFU
Eco
LALW
DRCM

(b)



(c)



(d)

Figure 5. 12: (a) Mean job time, (b) ENU, (c) ASU and (d) CE Usage for different replication algorithms, with different job schedulers.

5.5.3 Analysis: The Length of Access History

In this experiment the effect of access history length on the performance of DRCM and other existing algorithms is investigated and the dependency of DRCM on access history length. Using QAC scheduling algorithm and submitting 500 jobs to the grid, the access history length varying between 10^3 ms and 10^6 ms. In order to test the behavior of DRCM in different cases, namely when the access history has a poor information on file accesses, and when the access history has enough information on file accesses. Thus, we consider in this experiment that job submission rate (job delay) varying between 1000 ms and 2500 ms. The mean job time, ENU, and SE usage, are measured. The basic settings and parameters used in this experiment are shown in Table 5.14, and the summary of the results of this test is shown in Table 5.15

Table 5. 14: Access History test: parameters settings

Parameter	Value
Number of Jobs	500
Scheduler	QAC scheduler
Site Policy	All Job Types
Access history length	10^3 ms, 10^4 ms, 10^5 ms, 10^6 ms
Storage metric (D)	0.67
Max. Queue Size	200
Job Delay	1000 ms, 1500 ms, 2000 ms, 2500 ms

The first test in this experiment consider the access history length is 1000 ms and the job delay is 1000 ms, that means the access history contains information on only one

job files, i.e. the access history length is not enough as we have 500 jobs. However, the last test in this experiment considers the access history length is 10^6 ms and the job delay is 2500 ms, which means the access history has a good view of the overall access patterns.

Table 5. 15: Access History test: Simulation results

Access History Length	Job Delay	Metrics	LRU	LFU	Economy	LALW
10^3	1000	MJET	7543	7127	10995	11195
		ENU	30.96	30.85	37.26	46.26
		ASU	35.83	34.40	38.27	36.01
		CEU	21.49	20.25	23.53	25.31
10^4	1500	MJET	11518	9455	10980	10780
		ENU	43.18	43.13	38.26	44.26
		ASU	38.19	36.5	38.36	35.11
		CEU	24.30	18.35	23.38	24.95
10^5	2000	MJET	10816	10014	10769	9969
		ENU	41.74	47.87	38.26	35.26
		ASU	35.98	37.67	38.47	35.98
		CEU	23.75	19.40	24.80	25.19
10^6	2500	MJET	10011	9483	10991	9162
		ENU	46.49	47.56	37.26	33.26
		ASU	36.69	37.40	37.96	35.71
		CEU	18.86	20.30	24.15	25.91

From the simulation results shown in Table 5.15, the average of the values for each metric is computed to show the overall performance. The averages are shown in Table 5.16.

Table 5. 16: Access History test: Average results of the simulation

Metric	LRU	LFU	Eco	LALW	DRCM
MJET	9972	9020	10934	10277	9957
ENU	40.59	42.35	37.75	39.75	39.19
ASU	36.68	36.50	38.27	35.70	29.47
CEU	22.11	19.58	23.97	25.34	26.61

In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed by applying equation 5.1

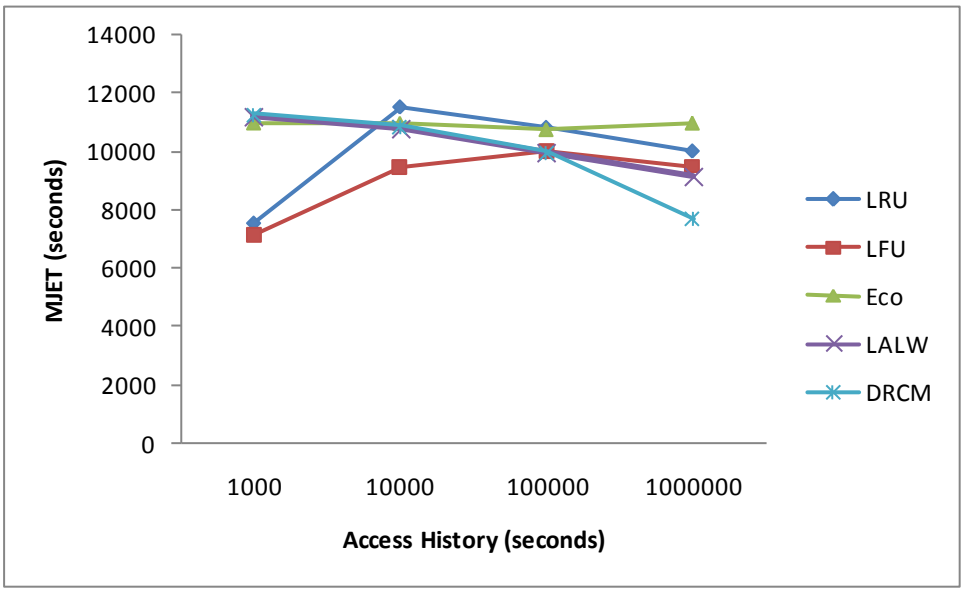
Table 6.17 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

Table 5. 17: Access History test: efficiency results

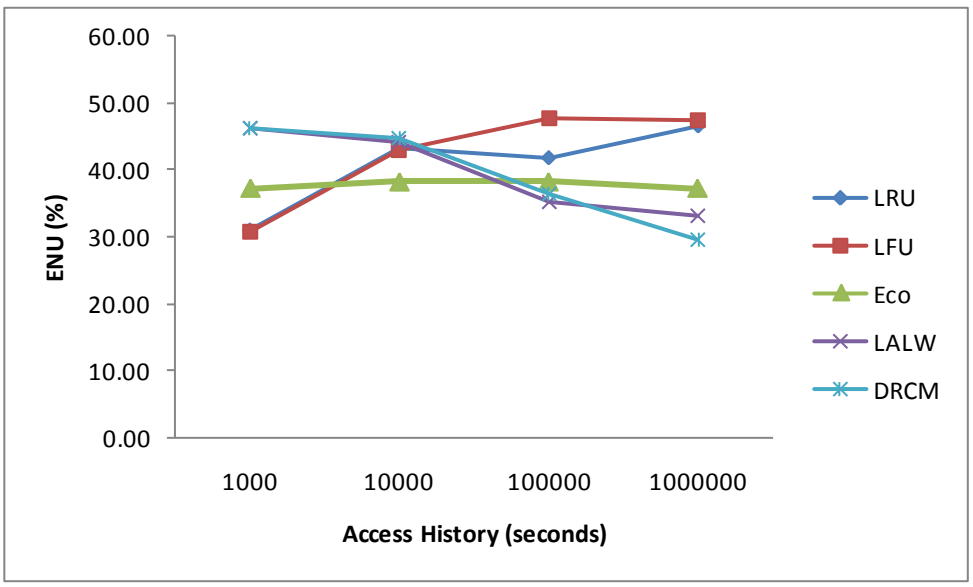
	LRU	LFU	Economy	LALW
MJET	0.14%	-10.40%	8.93%	3.10%
ENU	3.45%	7.46%	-3.81%	1.42%
ASU	19.66%	19.27%	23.01%	17.47%
CEU	20.38%	35.92%	11.03%	5.02%

Figure 5.13 is clearly shown that the performance of all algorithms get worse until the access history contains enough information on the files, and there is not a large variation in mean job time of each algorithm. The poor performance of Eco, LALW, and DRCM with small access histories, however, LRU and LRU are the best performer. LALW and DRCM perform badly with small access history because the files values changes rapidly that seemingly worthless files will be deleted when they

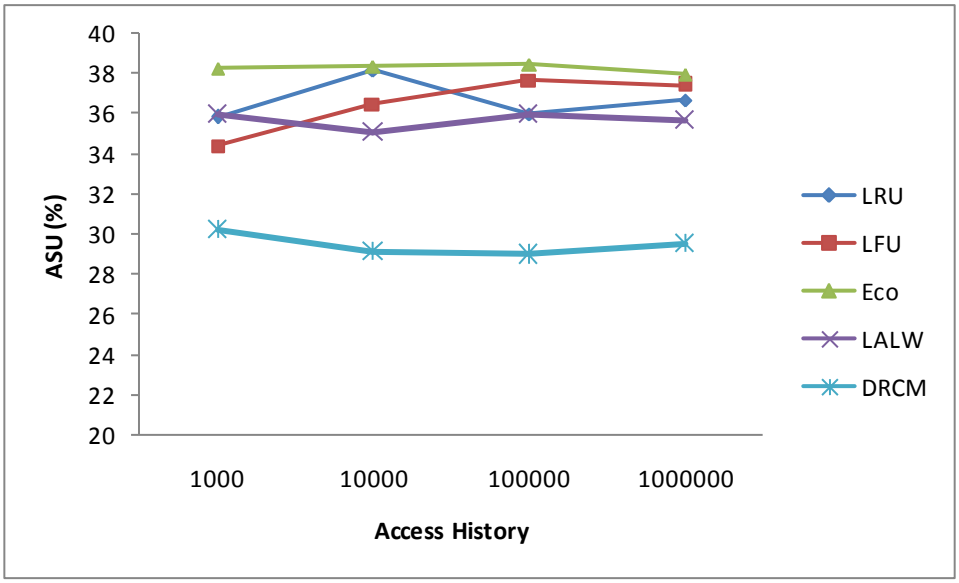
are likely to be requested in the near future. Those strategies namely LALW and DRCM require a large access history to be able to assess well which files are worth keeping. There is no noticeable effect of the length of access history on both of storage element usage and computing element usage.



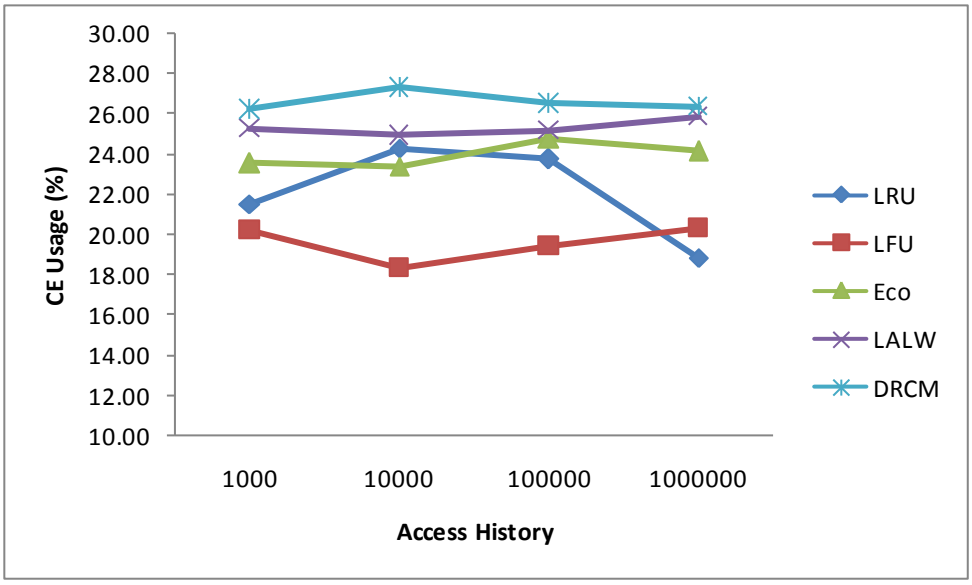
(a)



(b)



(c)



(d)

Figure 5. 13: (a) Mean job time, (b) ENU, (c) ASU and (d) CE Usage for different replication algorithms, with varying access history lengths.

5.5.4 Analysis: The Files and Storage Sizes

The sizes of the files and in turn the value of D may affect the performance of each replication algorithm. The less storage space available per site would lead to longer job times and larger effective network usage because fewer replicas can be accommodated in the grid. In this experiment we investigate the performance of DRCM with different sizes of the file varying from 200 MB to 2000 MB. The basic settings and parameters used in this experiment are shown in Table 5.18 and the summary of the results of this test is shown in Table 5.19.

Table 5. 18: File Size Test: Parameters Settings

Parameter	Value
Number of Jobs	500
Scheduler	QAC scheduler
Site Policy	All Job Types
Access history length	1000000 ms
Storage metric (D)	0.05, 0.37, 0.66, 1.31
Max. Queue Size	200
Job Delay	2500 ms

Table 5. 19: File Size Test: Simulation results

Storage Metric	Metrics	LRU	LFU	Economy	LALW	DRCM
0.05	MJET	11024	10721	10820	10821	10932
	ENU	89.21	88.91	90.56	87.34	88.39
	ASU	94.66	93.94	91.32	90.23	90.47
	CEU	26.05	23.22	25.19	24.38	26.01
0.37	MJET	10529	10281	11034	9586	9372

	ENU	55.19	53.22	56.92	51.48	47.57
	ASU	55.11	55.26	53.05	51.91	49.15
	CEU	25.31	24.93	25.33	23.81	25.81
	MJET	10011	9484	10992	9163	7692
0.66	ENU	46.49	47.56	37.26	33.77	30.17
	ASU	36.70	37.41	37.97	35.71	29.52
	CEU	24.53	25.78	25.44	25.52	26.19
	MJET	6712	7101	8973	7310	7287
1.31	ENU	13.8461	13.1587	15.7609	15.94	16.31
	ASU	22.92	19.65	22.54	19.21	18.73
	CEU	27.12	22.07	30.02	29.12	31.81

From the simulation results shown in Table 5.19, the average of the values for each metric is computed to show the overall performance. The averages are shown in Table 5.20.

Table 5. 20: File Size Test: Average of Simulation results

	LRU	LFU	Economy	LALW	DRCM
MJET	9569	9397	10455	9220	8821
ENU	51.18	50.71	50.13	47.13	45.61
ASU	52.35	51.57	51.22	49.27	46.97
CEU	25.75	24.00	26.50	25.71	27.46

In order to show the efficiency of the DRCM over the existing algorithms, the efficiency values are computed by applying equation 5.1

Table 5.21 shows the efficiency of the DRCM -as percentage values- over other existing algorithms.

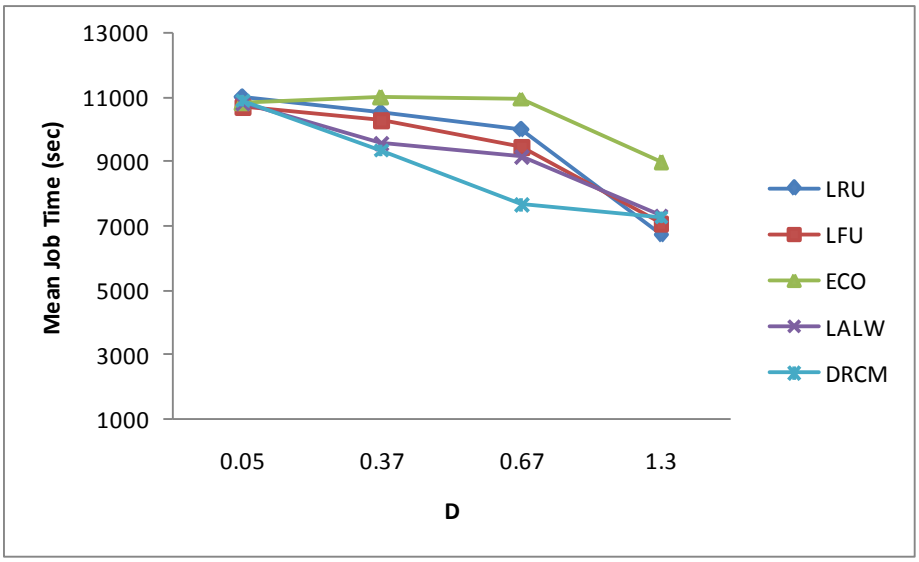
Table 5. 21: File Size Test: Efficiency results

	LRU	**LFU**	**Economy**	**LALW**
MJET	7.82%	6.13%	15.63%	4.33%
ENU	10.89%	10.06%	9.01%	3.23%
ASU	10.28%	8.92%	8.30%	4.66%
CEU	6.61%	14.39%	3.61%	6.79%

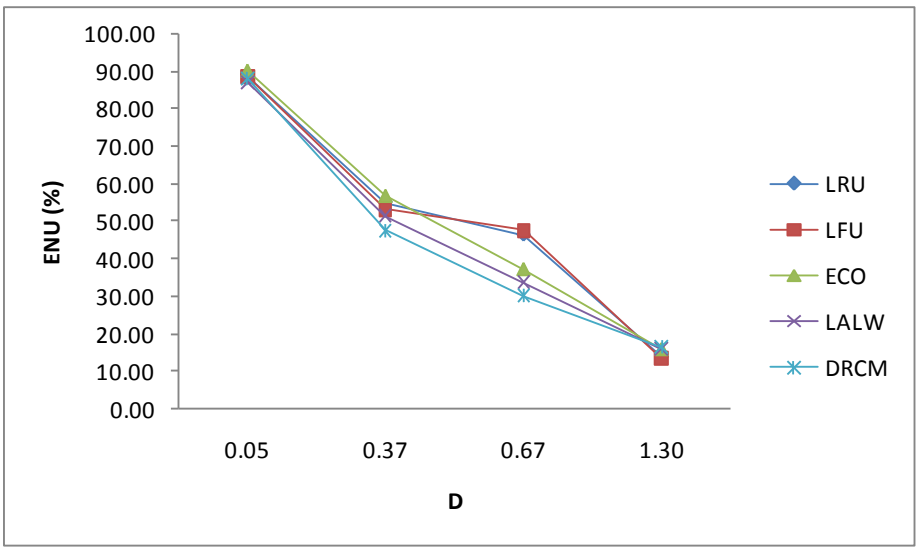
In what follows, we discuss and analyze the results that presented in Table 5.19. Firstly, for the smallest value of D (i.e. D=0.05) the mean job time for all strategies is high, as replication lose its advantage compared to remote access and each new job is more likely to request files which have not been requested before, because the available space in the storage elements is very limited. The job scheduler submits the jobs evenly among the sites, even if the sites have a heavy workload, and thus the increasing number of jobs that are waiting in the queue in the sites are increased as well as the mean job time. For the highest value of D (i.e. D = 1.31) as shown in Figure 5.14, the LRU and LFU are slightly faster than other strategies, because the files are likely to be read locally as LRU and LFU always replicate the files. Looking at ENU metric in Figure 5.14, it is noticeable that ENU falls as D decreases, for all replication algorithms due to the same reason, as there is enough space in storage elements to the extent that all of the replicas can be accommodated and read locally. The DRCM perform the best when (D= 0.37) and D = 0.66 as it gives the shortest job execution time and smallest value of ENU.

Looking at CE metric in Figure 5.14, there is a little variation in the computing element usage when value of D < 1. This is because the files are spread around the

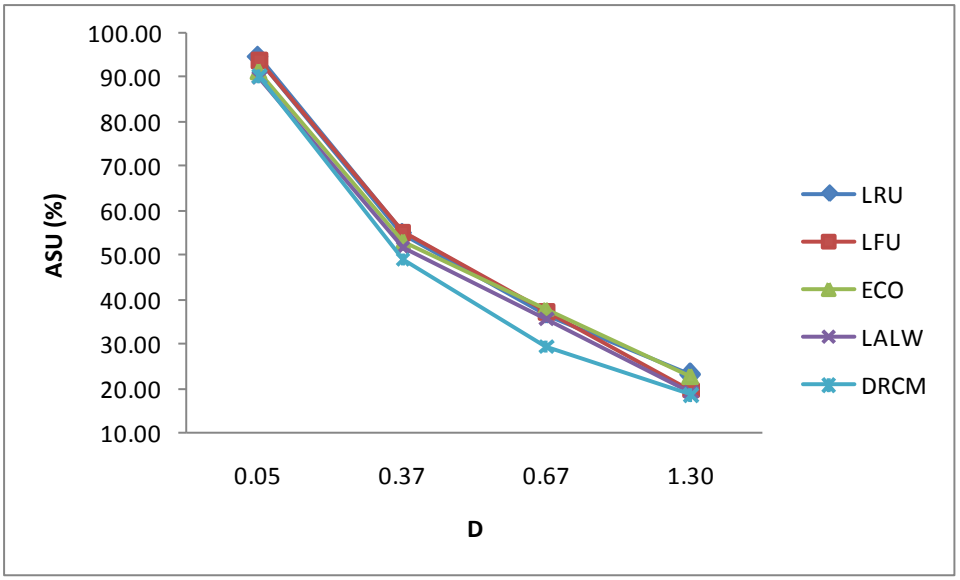
sites, in this case DRCM outperform other replication algorithms. When D > 1 (i.e. D=1.3) there is a noticeable increase in computing element usage meaning that replication algorithms lead to make balance in grid system. Due to large storage space that allows the sites to store all the files in grid system, therefore every site in grid is likely to be a candidate that chosen by job scheduler to run the job. As a result, job scheduler has more choices when submitting the jobs to the grid sites and can balance number of jobs at each site.



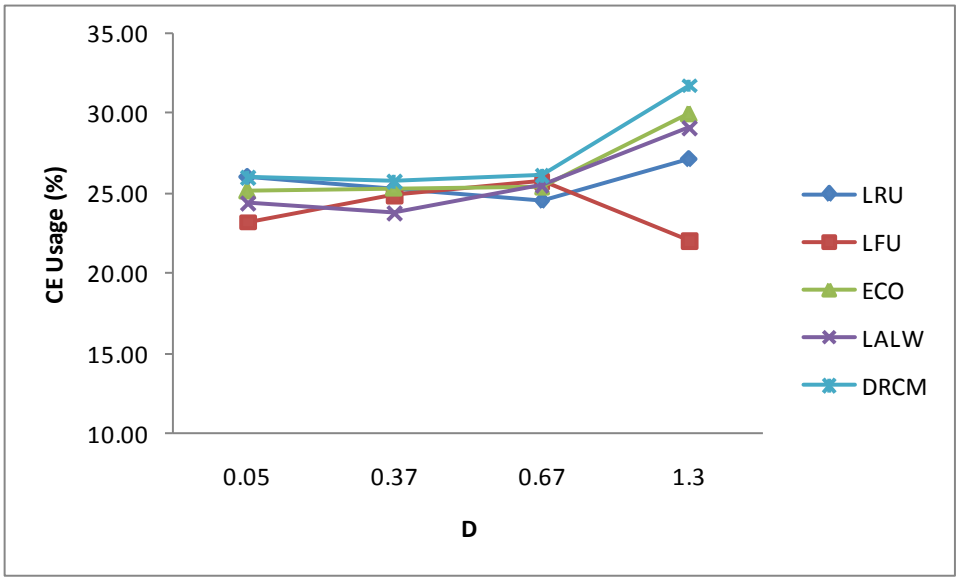
(a)



(b)



(c)



(d)

Figure 5 14: (a) Mean job time, (b) ENU, (c) ASU and (d) CE Usage for different replication algorithms, with varying storage element metric.

5.6 Summary of Chapter

In this chapter, we presented the result of our simulation experiments. In the simulation experiments, different scenarios were setup to evaluate the DRCM and other relevant replication algorithms. DRCM exhibits better performance in both dependent and independent data files simulation scenarios. The simulation results showed an overall improvement of the performance of data grid when using DRCM. As a result the overall bandwidth consumption decreased; moreover, DRCM does the best usage of storage resources. Besides that, DRCM greatly affects the work of job scheduler and in turn the overall computing element usage.

In the next chapter, the conclusions and contribution of the research work presented in this thesis will be stated. Then, some suggestions for further studies will be discussed.

CHAPTER SIX
CONCLUSION AND FUTURE RESEARCH WORK

6.1 Conclusion of the Research

This chapter presents a conclusion of the research work as explored and described in the thesis. The research contributions are supported by the experimental results which are highlighted. The applicability of the proposed algorithm in the real world is also presented, followed by a discussion of the research limitations. Eventually, several possible future research directions to realize and extend the work are also identified and recommended.

Data replication is a technique to move and cache data close to users. By replication, data access performance can be improved dynamically. The general idea of replication is to store copies of data in different locations so that data can be easily recovered if one copy at one location is lost or unavailable.

Therefore, the proposed algorithm (DRCM) has been designed and implemented as a response to the need of an enhanced replication algorithm in the established domain, where data proliferation and limited resources in data grids are common. The main problem that is addressed by this thesis is how to make a decision on replica creation and manage the newly created replicas in order to satisfy both the grid resources and grid users.

Resource satisfaction is achieved by reducing the overall cost which includes reducing storage cost and network bandwidth. On the other hand, user satisfaction is

achieved by reducing job execution time. This problem has been addressed by other researchers [47], but their work still needs enhancements, as discussed in Chapter 2 of this thesis.

DRCM allows for greater user satisfaction and resource satisfaction simultaneously because it complies with grid resource limitations and the requirements of the users' job. Making a decision on replication and deletion is not an easy task. It was observed that considering all grid request patterns in evaluating the files which will influence the decision, is better than considering only the most well-known request. In this context, the most well-known request is the request which is made directly by the user for a specific data file.

It was observed that considering the distribution of the replication sites along with other parameters such as the transfer time of data file among sites, and workload of each site have a significant effect on the overall system performance, specially the job execution time, because grid sites expose geographical localities in the data grid environment.

Deleting files from a storage element is not an easy decision, and thus the victim file must be selected carefully. It was observed that considering the value of the file combined with the file size in replica replacement strategy is better in enhancing the performance of the system than considering only file value, as in LFU and LRU policies.

6.2 Contribution of the Research

The major contribution of this research work is related to the proposing of a new replica creation algorithm that enhances the performance of the data grid by reducing job execution time and reducing the overall grid resource cost. Therefore, the thesis makes several important contributions, which are as follows:

i. A new replication decision strategy is proposed

In this research, the author has proposed a new strategy that decides on which replica should be deleted or created and how many replicas. Two mathematical models were used, namely the exponential growth/decay model and dependency model, for evaluating the files. The exponential model evaluates the files in terms of user tracking behavior and notes the importance of the files for users. However, the dependency model evaluates the files in terms of file tracking behavior. Then the results of the two models were combined together to form the file value that is used as an indicator to the importance of the file from the user perspective. On the other hand, the importance of the file from the system's perspective is computed using the current number of replicas that already exists in the data grid system. The algorithm makes the decision whether to increase the number of replicas of the most valuable files to face the high volume of requests, or to reduce the number of replicas of the less valuable files to save more storage space.

ii. A new replica placement strategy is proposed

The newly created replicas need to be hosted. So, the new replica placement strategy selects the best location sites that provide the minimum read cost. The read

cost is defined as the time required for transferring the replicas among the sites. The best location sites are determined by considering site workload, the places of exiting replicas, and the places of dependence files.

iii. A new replica replacement strategy is proposed

The replica replacement strategy (RRS) is invoked if the target storage at the site that has been chosen for placement of the newly created replica is full. The RRS determines the victim file to be replaced based on two parameters, namely file value and file size.

iv. The development of replica creation algorithm termed as DRCM is proposed

The main two key advantages of this proposed new algorithm (DRCM) are: firstly, the new algorithm embodies the above three strategies in one algorithm, which are considered as the essential functions in replication technique for the data grid. Secondly, the new algorithm performs the replication of multi-files in one decision making, i.e. when the DRCM decides to perform the replication or deletion process, the decision will include several files. Likewise, when selecting the candidate sites to host the newly created replicas, multi-locations would be selected to accommodate these multi-files.

v. The implementation of DRCM in a data grid simulator (OptorSim)

This implementation can be used by other researchers for comparison or modification purposes. It presents the core and the basic algorithm that includes the most common strategies involved in the replica creation stage.

6.3 Future works

The work reported in this research has opened up several avenues for exploration. From the previous in-depth discussion and detailed analysis, this research could be further improved and extended in several aspects. The potential improvement area in the thesis is summarized as follows:

There are plans to extend this replica creation algorithm (DRCM) to cover and include the replica management strategies, namely replica selection strategy and replica maintenance strategy. Replica selection strategy is responsible for finding the best replica location from among many replicas which are distributed across the data grid in order to provide the grid users with the required replicas in minimum response time, while they are running their jobs. In our algorithm used a simple replica selection strategy, which is already implemented in the simulator and this depends on the network bandwidth. Indeed, there are many criteria that should be considered in replica selection and since it plays a vital role in selecting the best replica location.

The second strategy that could be included in this algorithm is replica maintenance strategy. Due to the dynamic nature of the grid, candidate sites that hold the replicas currently may not be the best sites to fetch the replica. Therefore, there is a need to relocate the replicas to location sites that provide better services in the context of the current situation and network conditions.

REFERENCES

[1] J. P. M. G.A.Gravvanis, H.R. Arabina, D.A. Power, "Grid Technology and Applications: Recent Developments," New York: Nova Science Publishers, Inc., 2009.

[2] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," *San Francisco: Morgan Kaufmann Publishers*, vol. 24, p. 8, 1999.

[3] S. Venugopal, R. Buyya, and L. Winton, "A Grid service broker for scheduling e Science applications on global data Grids," *Concurrency and Computation: Practice and Experience*, vol. 18, pp. 685-699, 2006.

[4] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," *International Journal of Supercomputing Applications*, vol. 15, pp. 200-222, 2001.

[5] G. Wasson and M. Humphrey, "Policy and enforcement in virtual organizations," in *Proceedings of the 4th International Workshop on Grid Computing*, 2003, p. 125.

[6] I. Foster, "The grid enabling resource sharing within virtual organizations," in *WWW 2000 Conference*, 2002.

[7] Frederic Magoulès, *Fundamentals of grid computing: theory, algorithms and technologies*. USA: Chapman & Hall/CRC Numerical Analysis & Scientific Computing, 2010.

[8] A. Chervenak, E. Deelman, C. Kesselman, B. Allcock, I. Foster, V. Nefedova, J. Lee, A. Sim, A. Shoshani, and B. Drach, "High-performance remote access to climate simulation data: A challenge problem for data grid technologies," in *Super Computing*, 2003, pp. 1335-1356.

[9] I. Foster, E. Alpert, A. Chervenak, B. Drach, C. Kesselman, V. Nefedova, D. Middleton, A. Shoshani, A. Sim, and D. Williams, "The Earth System Grid II: Turning climate datasets into community resources," in *Annual Meeting of the American Meteorological Society*, 2002.

[10] B. Wilkinson, *Grid computing: techniques and applications*: Chapman & Hall/CRC, 2009.

[11] C. Nicholson, D. G. Cameron, A. T. Doyle, A. P. Millar, and K. Stockinger, "Dynamic data replication in lcg 2008," *Concurrency and Computation: Practice and Experience*, vol. 20, pp. 1259-1271, 2008.

[12] A. Chervenak, E. Deelman, I. Foster, W. Hoschek, A. Iamnitchi, C. Kesselman, M. Ripeanu, B. Schwartzkopf, H. Stockinger, and B. Tierney, "Giggle: A framework for constructing scalable replica location services," in *International IEEE Supercomputing Conference (SC 2002)* Baltimore, USA, 2002, pp. 1-17.

[13] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke., "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," *Journal of Network and Computer Applications*, vol. 23, 2001.

[14] L. Guy, P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger, "Replica management in data grids," in *Global Grid Forum*. vol. 5, 2002.

[15] H. Lamehamedi, Z. Shentu, B. Szymanski, and E. Deelman, "Simulation of dynamic data replication strategies in data grids," in *Proceedings of 12th Heterogeneous Computing Workshop (HCW2003)*, Nice, France, , 2003.

[16] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman, "Data Replication Strategies in Grid Environments," in *Fifth International Conference on Algorithms and Architectures for Parallel Processing*, 2002, p. p.378.

[17] E. Otoo, F. Olken, and A. Shoshani, "Disk cache replacement algorithm for storage resource managers in data grids," in *2002 ACM/IEEE conference on Supercomputing*, Baltimore, Maryland 2002, pp. 1-15.

[18] K. Ranganathan and I. Foster, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid," *International Grid Computing Workshop*, pp. 75-86, 2001.

[19] X. You, G. Chang, X. Chen, C. Tian, and C. Zhu, "Utility-Based Replication Strategies in Data Grids," in *Fifth International Conference on Grid and Cooperative Computing*, 2006, pp. 500-507.

[20] M. Tang, B. S. Lee, X. Tang, and C. K. Yeo, "The impact of data replication on job scheduling performance in the Data Grid," *Future Generation Computer Systems*, vol. 22, pp. 254-268, 2006.

[21] Srikummar Venugopal, "Scheduling Distributed Data-Intensive Applications on Global Grids," PhD thesis, University of Melbourne, Australia, 2006.

[22] K. Ranganathan and I. Foster, "Design and Evaluation of Dynamic Replication Strategies for a High Performance Data Grid," in *International Conference on Computing in High Energy and Nuclear Physics*, Beijing, 2001.

[23] R. S. Chang and P. H. Chen, "Complete and fragmented replica selection and retrieval in Data Grids," *Future Generation Computer Systems,* vol. 23, pp. 536-546, 2007.

[24] M. Tang, B. S. Lee, C. K. Yeo, and X. Tang, "Dynamic replication algorithms for the multi-tier Data Grid," *Future Generation Computer Systems,* vol. 21, pp. 775-790, 2005.

[25] R. M. Rahman, K. Barker, and R. Alhajj, "Replica placement strategies in data grid," *Journal of Grid Computing,* vol. 6, pp. 103-123, 2008.

[26] R. M. Rahman, R. Alhajj, and K. Barker, "Replica selection strategies in data grid," *Journal of Parallel and Distributed Computing,* vol. 68, pp. 1561-1574, 2008.

[27] R. M. Rahman, K. Barker, and R. Alhajj, "A Predictive Technique for Replica Selection in Grid Environment," in *Seventh IEEE International Symposium on Cluster Computing and the Grid CCGRID 2007*, 2007, pp. 163-170.

[28] R. M. Ken, K. Barker, and R. Alhajj, "Study of different replica placement and maintenance strategies in data grid," 2007, pp. 171-178.

[29] R. M. Rahman, K. Barker, and R. Alhajj, "Performance evaluation of different replica placement algorithms," *International Journal of Grid and Utility Computing,* vol. 1, pp. 121-133, 2009.

[30] Abdelsalam A. Helal, Abdelsalam A. Heddaya, and Bharat B. Bhargava, *Replication techniques in distributed systems*: Kluwer Academic Publishers, 1996.

[31] Caitriana M. Nicholson, "File management for HEP data grids," PhD thesis, University of Glasgow, 2006.

[32] M. Xie, Y. S. Dai, and K. L. Poh, *Computing systems reliability: models and analysis*: Springer Us, 2004.

[33] H. Lamehamedi and B. K. Szymanski, "Decentralized data management framework for data grids," *Future Generation Computer Systems,* vol. 23, pp. 109-115, 2007.

[34] C. T. Yang, C. J. Huang, and T. C. Hsiao, "A Data Grid File Replication Maintenance Strategy Using Bayesian Networks," in *Intelligent Systems Design and Applications, 2008. ISDA'08*, 2008.

[35] H. H. E. Al Mistarihi and C. H. Yong, "Replica management in data grid," *International Journal of Computer Science and Network Security IJCSNS,* vol. 8, p. 22, 2008.

[36] M. Tang, B. Lee, X. Tang, and C. Yeo, "Combining data replication algorithms and job scheduling heuristics in the data grid," *Lecture notes in computer science*, vol. 3648, p. 381, 2005.

[37] C. Ruay-Shiung, C. Hui-Ping, and W. Yun-Ting, "A dynamic weighted data replication strategy in data grids," in *AICCSA 2008: Proceedings of IEEE/ACS International Conference on computer systems and applications*, 2008, pp. 414-421.

[38] H. P. Chang, "A Dynamic Data Replication Strategy Using Access-Weights in Data Grids," 2006.

[39] C. Wang, C. Yang, and M. Chiang, "A Fair Replica Placement for Parallel Download on Cluster Grid," *Lecture Notes in Computer Science*, vol. 4658, p. 268, 2007.

[40] C. T. Yang, C. P. Fu, and C. J. Huang, "A dynamic file replication strategy in data grids," in *TENCON 2007-2007 IEEE Region 10 Conference*, 2007, pp. 1-5.

[41] Q. Rasool, L. Jianzhong, G. S. Oreku, Z. Shuo, and Y. Donghua, "A load balancing replica placement strategy in Data Grid," in *Proceedings of Third International Conference on Digital Information Management, ICDIM*, London, UK, 2008, pp. 751-756.

[42] M. Shorfuzzaman, P. Graham, and R. Eskicioglu, "Popularity-Driven Dynamic Replica Placement in Hierarchical Data Grids," in *Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008*, 2008, pp. 524-531.

[43] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in *Global and Peer-to-Peer Computing on Large Scale Distributed Systems Workshop*, 2002, pp. 376–381.

[44] L. Yi-Fang, L. Pangfeng, and W. Jan-Jan, "Optimal placement of replicas in data grid environments with locality assurance," in *Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on*, 2006, p. 8.

[45] L. Pangfeng and W. Jan-Jan, "Optimal replica placement strategy for hierarchical data grid systems," in *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, 2006, p. 4 pp.

[46] Y. Mansouri, M. Garmehi, M. Sargolzaei, and M. Shadi, "Optimal Number of Replicas in Data Grid Environment," in *First International Conference on Distributed Framework and Applications, 2008. DFmA 2008.* , 2008, pp. 96-101.

[47] David G. Cameron, "Replica management and optimisation for data grids," PhD. Thesis, University of Glasgow, 2005.

[48] W. B. David, "Evaluation of an economy-based file replication strategy for a data grid," in *International Workshop on Agent based Cluster and Grid Computing*, 2003, pp. 120–126.

[49] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project," *Lecture Notes in Computer Science*, pp. 77-90, 2000.

[50] J. H. Abawajy, "Placement of file replicas in data grid environments," *Lecture Notes in Computer Science*, pp. 66-73, 2004.

[51] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data management and transfer in high-performance computational grid environments," *Parallel Computing*, vol. 28, pp. 749-771, 2002.

[52] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S. Tuecke, and I. Foster, "Secure, efficient data transport and replica management for high-performance data-intensive computing," in *IEEE Mass Storage Systems and Technologies*, 2001.

[53] I. Foster, "The Grid: A New Infrastructure for 21st Century Science," *PHYSICS TODAY*, vol. 55, pp. 42-47, 2002.

[54] W. E. Johnston, "Computational and data Grids in large-scale science and engineering," *Future Generation Computer Systems*, vol. 18, pp. 1085-1100, 2002.

[55] P. Avery, "Data Grids: a new computational infrastructure for data-intensive science," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 360, p. 1191, 2002.

[56] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer*, vol. 35, pp. 37-46, 2002.

[57] S. Shen, *Grid computing: International Symposium on Grid Computing*: Springer-Verlag New York Inc, 2008.

[58] F. Magoulès, J. Pan, K. A. Tan, and A. Kumar, *Introduction to grid computing* vol. 6: CRC, 2009.

[59] F. Gagliardi, B. Jones, F. Grey, M. E. Bégin, and M. Heikkurinen, "Building an infrastructure for scientific Grid computing: status and goals of the EGEE project," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 363, p. 1729, 2005.

[60] T. Hey and A. E. Trefethen, "Cyberinfrastructure for e-Science," *Science,* vol. 308, p. 817, 2005.

[61] The LHCb Collaboration. LHCb Computing Model. Technical Report CERN-LHCC-2004-036/G-084, CERN, January 2005.

[62] F. Berman, G. Fox, and T. Hey., *The Grid: Past, Present, Future, Grid Computing: Making the Global Infrastructure a Reality*. London, UK: Wiley Press, 2003.

[63] G. Fox, S. H. Ko, M. Pierce, O. Balsoy, J. Kim, S. Lee, K. Kim, S. Oh, X. Rao, and M. Varank, "Grid services for earthquake science," *Concurrency and Computation: Practice and Experience,* vol. 14, pp. 371-393, 2002.

[64] I. Foster, "Internet computing and the emerging grid," *Nature Web Matters,* vol. 7, 2000.

[65] N. Kelly, P. V. Jithesh, D. R. Simpson, P. Donachy, T. J. Harmer, R. Perrott, J. Johnston, P. Kerr, M. McCurley, and S. McKee, "Bioinformatics data and the grid: The GeneGrid data manager," in *UK e-Science All Hands Meeting 2004 (AHM04)*, 2004.

[66] F. Magoulès and L. Yu, *Grid resource management: towards virtual and services compliant grid computing*: CRC Press, 2009.

[67] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing," *ACM Sigmod Record,* vol. 34, pp. 44-49, 2005.

[68] High Energy Physics Experiment Website, "http://www.hep.net."

[69] European Organization for Nuclear Research (CERN), "http://public.web.cern.ch/Public/Welcome.html."

[70] The ALICE Collaboration. ALICE Computing Model. Technical Report CERN-LHCC-2004-038/G-086, CERN, January 2005.

[71] The ATLAS Collaboration. The ATLAS Computing Model. Technical Report CERN-LHCC-2004-037/G-085, CERN, January 2005.

[72] The CMS Collaboration. The CMS Computing Model. Technical Report CERN-LHCC-2004-035/G-083, CERN, January 2005.

[73] K. Holtman, "CMS data grid system overview and requirements," *CMS Note*, vol. 37, July 2001.

[74] CMS Data Challenge 2004, " http://www.uscms.org/s&c/dc04/."

[75] The MONARC Project. Models of Networked Analysis at Regional Centres for LHC Experiments (MONARC) Phase 2 Report. Technical Report CERN/LCB 2000-001, CERN, March 2000.

[76] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, and P. Fox, "The earth system grid: Supporting the next generation of climate modeling research," *Proceedings of the IEEE*, vol. 93, pp. 485-495, 2005.

[77] Sloan Digital Sky Survey website. Available online at: http://www.sdss.org/

[78] R. Dooley, K. Milfeld, C. Guiang, S. Pamidighantam, and G. Allen, "From proposal to production: Lessons learned developing the computational chemistry grid cyberinfrastructure," *Journal of Grid Computing*, vol. 4, pp. 195-208, 2006.

[79] U. Farooq, S. Majumdar, and E. W. Parsons, "Engineering grid applications and middleware for high performance," in *Proceedings of the 6th international workshop on Software and performance*, 2007, pp. 141-152.

[80] R. Moore, T. A. Prince, and M. Ellisman, "Data-intensive computing and digital libraries," *Communications of the ACM*, pp. 56–62, 1998.

[81] J. Singh, N. Kaur, and R. Singh, "The Uniqueness of Data Grid Over Other Data-Intensive Paradigms," *Enterprise information systems in twenty-first century*, p. 334, 2009.

[82] C. E. Palau, J. C. Guerri, M. Esteve, F. Carvajal, and B. Molina, "CCDN: campus content delivery network learning facility," in *IEEE International Conference on Advanced Learning Technologies (ICALT'03)* 2003.

[83] J. P. Mulerikkal, "An Architecture for Distributed Content Delivery Network," in *15th IEEE International Conference on Networks, ICON*, 2007, pp. 359-364.

[84] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (IMW '01),*, San Francisco, CA, USA. ACM Press, New York, NY, USA., 2001, pp. 169-182.

[85] Akamai, "http://www.akamai.com."

[86] S. Ceri and G. Pelagatti, *Distributed databases principles and systems*. McGraw-Hill, New York, USA, 1984.

[87] Ozsu and P. Valduriez, "Principles of distributed database systems," *Alan Apt, New Jersey*, 1999.

[88] S. Venugopal, R. Buyya, and K. Ramamohanarao, "A taxonomy of data grids for distributed data sharing, management, and processing," *ACM Computing Surveys (CSUR)*, vol. 38, p. 3, 2006.

[89] S. Goel and R. Buyya, "Data Replication Strategies in Wide Area Distributed Systems," *Enterprise Service Computing: From Concept to Deployment*, pp. 211-241, 2006.

[90] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. Van Steen, "Replication for web hosting systems," *ACM Computing Surveys (CSUR)*, vol. 36, pp. 291-334, 2004.

[91] M. R. Rahman, "Replica placement and selection strategies in data grids," in *Department of Computer Science*. vol. PhD. thesis Alberta: University of Calgary, 2007.

[92] Stockinger and H, "Distributed database management systems and the data grid," in *Proceedings of Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, MSS'01*, San Diego, CA, 2001.

[93] M. Di Stefano and J. Wiley, *Distributed data management for grid computing*: Wiley Online Library, 2005.

[94] M. Carman, F. Zini, L. Serafini, and K. Stockinger, "Towards an economy-based optimisation of file access and replication on a data grid," in *Proceedings of Second IEEE International Symposium on Cluster Computing and the Grid (CCGRID'02)*, 2002, p. 340.

[95] L. Dutka, R. Slota, D. Nikolow, and J. Kitowski, "Optimization of Data Access for Grid Environment," in *Grid Computing*, 2004, pp. 93-102.

[96] Mathew J. Wyatt, Nigel G.D. Sim, Dianna L. Hardy, and I. M. Atkinson, "YourSRB: A cross platform interface for SRB and Digital Libraries," in *Proceedings of the fifth Australasian symposium on ACSW frontiers-Volume 68*, 2007, pp. 79-85.

[97] S. Krishnamurthy, W. H. Sanders, and M. Cukier, "Performance evaluation of a probabilistic replica selection algorithm," in *Proceedings of the Seventh*

International Workshop on Object-Oriented Real-Time Dependable Systems, (WORDS 2002). , 2002, pp. 119-127.

[98] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S. Y. Chen, and R. Olschanowsky, "Storage resource broker-managing distributed data in a grid," *Computer Society of India Journal, special issue on SAN,* vol. 33, pp. 42-54, 2003.

[99] O. Othman, C. O'Ryan, and D. Schmidt, "The Design and Performance of an Adaptive CORBA Load Balancing Service," *IEEE Distributed Systems Online,* vol. 2, pp. 48-60, 2001.

[100] S. Vazhkudai, S. Tuecke, and I. Foster, "Replica selection in the globus data grid," in *Proceedings of International Workshop on Data Models and Databases on Clusters and the Grid (DataGrid 2001)*, 2001, pp. 106–113.

[101] The Globus Alliance, "http://www.globus.org/."

[102] S. B. Karl, K. Czajkowski, S. Fitzgerald, I. Foster, A. Johnson, C. Kesselman, J. Leigh, and S. Tuecke, "Application Experiences with the Globus Toolkit," in *Proceedings of Eighth IEEE Symposium on High Performance Distributed Computing*, 1998, pp. 81-89.

[103] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Principles*: Wiley India Pvt. Ltd., 2006.

[104] S. Y. Ko, R. Morales, and I. Gupta, "New worker-centric scheduling strategies for data-intensive grid applications," in *Proceedings of the 8th ACM/IFIP/USENIX international conference on Middleware*, 2007, pp. 121-142.

[105] L. Meyer, J. Annis, M. Wilde, M. Mattoso, and I. Foster, "Planning spatial workflows to optimize grid performance," in *Proceedings of the 2006 ACM symposium on Applied computing*, 2006, pp. 786-790.

[106] Q. Rasool, J. Li, and S. Zhang, "Replica Placement in Multi-tier Data Grid," in *Proceedings of 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp. 103-108.

[107] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "Evaluating scheduling and replica optimisation strategies in OptorSim," *Journal of Grid Computing,* pp. 57-69, March 2004.

[108] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, "Simulation of Dynamic Grid Replication Strategies in OptorSim," *Journal of High Performance Computing Applications,* vol. 17, 2003.

[109] F. Ben Charrada, H. Ounelli, and H. Chettaoui, "An Efficient Replication Strategy for Dynamic Data Grids," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pp. 50-54.

[110] F. Ben Charrada, H. Ounelli, and H. Chettaoui, "An Efficient Replication Strategy for Dynamic Data Grids," in *Proceedings of International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC),*, 2010, pp. 50-54.

[111] Z. Wuqing, X. Xianbin, W. Zhuowei, Z. Yuping, and H. Shuibing, "A Dynamic Optimal Replication Strategy in Data Grid Environment," in *Proceedings of International Conference on Internet Technology and Applications, ,* 2010, pp. 1-4.

[112] H. Zhong, Z. Zhang, and X. Zhang, "A Dynamic Replica Management Strategy Based on Data Grid," in *Proceedings of 2010 Ninth International Conference on Grid and Cloud Computing*, 2010, pp. 18-23.

[113] K. Sashi and A. S. Thanamani, "A New Replica Creation and Placement Algorithm for Data Grid Environment," in *Proceedings of 2010 International Conference on Data Storage and Data Engineering*, 2010, pp. 265-269.

[114] M. Teng and L. Junzhou, "A prediction-based and cost-based replica replacement algorithm research and simulation," in *Proceedings of 19th International Conference on Advanced Information Networking and Applications, (AINA 2005). ,* 2005, pp. 935-940.

[115] T. Tian and J. Luo, "A Prediction-based Two-Stage Replica Replacement Algorithm," in *Proceedings of 11th International Conference on Computer Supported Cooperative Work in Design, (CSCWD 2007)*, 2007, pp. 594-598.

[116] T. Tian and J. Luo, "A VO-Based Two-Stage Replica Replacement Algorithm," *Network and Parallel Computing,* pp. 41-50, 2010.

[117] W. Zhao, X. Xu, N. Xiong, and Z. Wang, "A Weight-Based Dynamic Replica Replacement Strategy in Data Grids," in *Proceedings of Asia-Pacific Services Computing Conference, ,* 2009, pp. 1544-1549.

[118] S. M. Park, J. H. Kim, Y. B. Ko, and W. S. Yoon, "Dynamic data grid replication strategy based on Internet hierarchy," *International Workshop on Grid and Cooperative Computing,* vol. 1001, pp. 1324–1331, 2004.

[119] M. Garmehi and Y. Mansouri, "Optimal Placement Replication on Data Grid Envirments," in *Proceedings of Information Technology, (ICIT 2007). 10th International Conference on*, 2007, pp. 190-195.

[120] K. Kalpakis, K. Dasgupta, and O. Wolfson, "Optimal placement of replicas in trees with read, write, and storage costs," *IEEE Transactions on Parallel and Distributed Systems*, pp. 628-637, 2001.

[121] M. Bsoul, A. Al-Khasawneh, Y. Kilani, and I. Obeidat, "A threshold-based dynamic data replication strategy," *The Journal of Supercomputing*, pp. 1-10, 2010.

[122] R. M. Rahman, K. Barker, and R. Alhajj, "Replica placement in data grid: considering utility and risk," in *Proceedings of Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, 2005.

[123] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management science*, pp. 1861-1871, 2004.

[124] Y. F. Lin, J. J. Wu, and P. Liu, "A List-Based Strategy for Optimal Replica Placement in Data Grid Systems," in *Proceedings of Parallel Processing, 2008. ICPP'08. 37th International Conference on*, 2008, pp. 198-205.

[125] S. Naseera and K. V. M. Murthy, "Agent Based Replica Placement in a Data Grid Environement," in *Proceedings of First International Conference on Computational Intelligence, Communication Systems and Networks. CICSYN'09.*, 2009, pp. 426-430.

[126] Z. Challal and T. Bouabana-Tebibel, "A priori replica placement strategy in data grid," in *Proceedings of 2010 International Conference on Machine and Web Intelligence (ICMWI)*, , 2010, pp. 402-406.

[127] R. M. Rahman, K. Barker, and R. Alhajj, "Replica selection in grid environment: a data-mining approach," in *Proceedings of the 2005 ACM symposium on Applied computing*, 2005, pp. 695-700.

[128] S. Vazhkudai, "Enabling the co-allocation of grid data transfers," in *Proceedings of 4th International Workshop on Grid Computing*, Phoenix, Arizona, USA, , 2003, pp. 44-51.

[129] Y. Zhao and Y. Hu, "GRESS–a grid replica selection service," in *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS-2003)*, 2003.

[130] T. Ho and D. Abramson, "The griddles data replication service," in *Proceedings of First International Conference on e-Science and Grid Computing*, 2005, pp. 278 - 286.

[131] C. Ferdean and M. Makpangou, "A scalable replica selection strategy based on flexible contracts," in *Proceedings of the Third IEEE Workshop on Internet Applications (WIAPP'03)*, Washington, DC, USA,, 2003, pp. 95 - 99.

[132] B. M. E. Moret and H. D. Shapiro, "Algorithms and experiments: The new (and old) methodology," *Journal of Universal Computer Science*, vol. 7, pp. 434-446, 2001.

[133] R. Wolski, "Forecasting network performance to support dynamic scheduling using the network weather service," in *Proceedings of The Sixth IEEE International Symposium on High Performance Distributed Computing*, 1997, pp. 316-325.

[134] W. Dubitzky and I. ebrary, *Data mining techniques in grid computing environments*: Wiley Online Library, 2008.

[135] MONORAC Project, "http://cern.ch/MONORC."

[136] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1175-1220, 2002.

[137] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," in *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing, 2002. HPDC-11 2002.* , 2002, pp. 352-358.

[138] Grid Datafarm, "http://datafarm.apgrid.org."

[139] Rohaya Latip, "Data Replication with 2D Mesh Protocol for Data Grid," in *Faculty of Computer Science and Information Technology*. vol. PhD. thesis: Universiti Putra Malaysia, 2009.

[140] M. Radi, "Maintaining Replica Consistency Over Large-Scale Data Grid Using Update Propagation Technique," in *Faculty of Computer Science and Information Technology*. vol. PhD: Universiti Putra Malaysia, 2009.

[141] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini, "Analysis of scheduling and replica optimisation strategies for data grids using OptorSim," *Journal of Grid Computing*, vol. 2, pp. 57-69, 2004.

[142] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini, "Optorsim: A grid simulator for studying dynamic data replication strategies," *International Journal of High Performance Computing Applications*, vol. 17, pp. 403-416, 2003.

[143] The European Data Grid Project, " http://eudatagrid.web.cern.ch/eu-datagrid/."

[144] H. H. E. Al-Mistarihi and C. H. Yong, "Response Time Optimization for Replica Selection Service in Data Grids," *Journal of Computer Science*, vol. 4, pp. 487-493, 2008.

[145] L. Hong, Q. Xue-dong, L. Xia, L. Zhen, and W. Wen-xing, "Fast Cascading Replication Strategy for Data Grid," in *Proceedings of the 2008 International Conference on Computer Science and Software Engineering-Volume 03*, 2008, pp. 186-189.

[146] M. Lei, S. V. Vrbsky, and X. Hong, "A dynamic data grid replication strategy to minimize the data missed," in *Proceedings of 3rd International Conference on Broadband Communications, Networks and Systems. BROADNETS.* , 2007, pp. 1-10.

[147] B. T. Huffman, R. McNulty, T. Shears, R. S. Denis, and D. Waters, "The CDF/D0 UK GridPP Project," *CDF Internal Note*, vol. 5858, 2002.

[148] D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "UK grid simulation with OptorSim," in *Proceedings of UK e-Science All Hands Meeting*, Nottingham, UK, 2003.

[149] R. G. Sargent, "Verification and validation of simulation models," in *proceedings of Winter Simulation Conference*, 2005, pp. 130-143.

[150] D. Minoli, *A networking approach to grid computing*. USA.: Wiley-Interscience, 2005.

[151] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," *Grid Computing—GRID 2001*, pp. 75-86, 2001.

[152] S. P. Kapitza, "The statistical theory of global population growth," in *Formal descriptions of developing systems*, 2003, p. 35.

[153] M. Kremer, "Population growth and technological change: one million BC to 1990," *The Quarterly Journal of Economics*, vol. 108, pp. 681-716, 1993.

[154] J. U. Kreft, G. Booth, and J. W. T. Wimpenny, "BacSim, a simulator for individual-based modelling of bacterial colony growth," *Microbiology*, vol. 144, p. 3275, 1998.

[155] A. D. MacCormack, J. Rusnak, and C. Y. Baldwin, *Exploring the structure of complex software designs: An empirical study of open source and proprietary code*: Citeseer, 2004.

[156] M. Hassan and R. Jain, *High performance TCP/IP networking*. United States of America: Pearson Prentice Hall, 2004.

Appendix A

Summary of existing work in replica placement

Authors	Technique	Variable	Methodology
Chang-Ming Xing [126]	MinimizeDelay MaxmimizeDelayDiff MinimizeMaxDelay	▪ Network Bandwidth ▪ Request Time	Replica is placed on the node that can reduce delay the most
Chih-Ming Wang et al. [40]	Fair Replica Placement	▪ Bandwidth ▪ Number of user request	Duplicate the file to a node that provides minimum average transmission cost (maximum bandwidth)
Qaisar Rasool et al. [42]	Fair Share Replication (FSR)	▪ Access load ▪ Storage load	The decision of replica placement is made based on the access load and the storage load of the candidate replica servers and their sibling nodes
Kavitha Ranganathan [44]	Dynamic Placement Algorithm	▪ Storage Cost ▪ Transfer Time	Maximize the deference between replication benefit and replication cost
Yi-Fang Lin [123]	List-based Strategy	▪ Workload (Storage + Request Time)	Determine the optimal locations for placing a given number of replicas so that the maximum server workload is minimized, and all requests are

			satisfied
Mehran Garmehi et al. [118]	Optimal Placement of Replicas	▪ Communication/ read Cost ▪ Storage Cost	Place the replica so that the overall cost (read and storage cost) is minimized
Chao-Tung Yang et al. [41]	Dynamic Maintenance Services (DMS)	▪ Request Frequency ▪ Storage Capacity	If the request frequency of file is more than the maximum request rate, and there is free space in the site then DMS will duplicate file to that location
Ruay-Shing et al. [2]	Latest Access Largest Weight (LALW)	▪ Access Frequency	According to the access frequencies for each file that has been requested, a popular file is replicated to a suitable site
Kavitha and Foster [24]	Best client	▪ Request Time	A replica created at node that generates maximum requests
	Cascading		Replica trickles down to lower tier if number of requests exceeds given threshold
	Caching		A requesting client receives and stores a copy locally
	Fast Spread		A replica is stored at each node along the

			path toward client
Yi-Fang Lin et al. [45]	MinMaxLoad	▪ User Request	Given the number of replicas, find the locations so that the maximum workload is minimized
Rahman et al. [121]	MinimizeExpectedUtil	▪ Distance (Response Time) ▪ File Requests	The node with lowest expected utility is selected to host the replica
	MaximizeTimeDiffUtil		The node with maximum time difference utility is selected to host the replica
	MinimizeMaxRisk		The replica is placed at the site with maximum risk index
	MinimizeMaxAvgRisk		The replica is placed at the site with highest average index
Liang Hong et al. [127]	Fast Cascading Replication	▪ Number of Access ▪ Storage Space	Once the number of access exceeds a threshold, and there is enough space, a replica is created at the next level